

Laksh Bhatia

Distributed Cyber-Physical Systems with Unmanned Aerial Vehicles

School of Science

Thesis submitted for examination for the degree of Master of
Science in Technology.

Espoo 31.7.2017

Thesis supervisor:

Vesa Hirvisalo, Senior University Lecturer

Thesis advisor:

Dr. David Boyle, Imperial College London

Author: Laksh Bhatia

Title: Distributed Cyber-Physical Systems with Unmanned Aerial Vehicles

Date: 31.7.2017

Language: English

Number of pages: 11+76

Degree Programme in Computer Science and Engineering
Master's Programme in ICT Innovation - Embedded Systems

Supervisor: Vesa Hirvisalo, Senior University Lecturer

Advisor: Dr. David Boyle, Imperial College London

This thesis discusses the merger between Unmanned Aerial Vehicles (UAVs) and static wireless sensor networks (WSNs). It explores and demonstrates the use of UAVs as mobile sinks to collect data from static sensor networks. A communication protocol is developed to approach a 100% data reliability while trying to maximise the speed of the UAV.

An energy model and speed versus reliability models are developed and tested using MATLAB. Mathematical models are developed to calculate the energy needed by nodes in such a system. The energy model developed is used to inform the design of recharging systems with wireless power transfer and consider energy harvesting opportunities.

The protocol developed is an asynchronous communication protocol. It is developed in ContikiOS on top of ContikiMAC radio duty cycling protocol using the Rime communication stack. A series of indoor and outdoor tests are conducted using real hardware and the performance of this protocol is compared with CTP.

The results show that the protocol developed has 100% data reliability when the speed of the UAV is less than 12m/s. Based on the performance results obtained, subsequent numerical analysis shows that operational lifetime of nodes under these conditions can extend to 1.8 years using a typical 2400 mAH battery. This work is one of the first practical demonstrations of UAVs with WSN and highlights a number of consequential research questions.

Keywords: CPS, UAV, WSNs, Contiki, ContikiMAC, TSCH, Windfarms, Oil and Gas Pipelines, Collect, 100% data reliability, speed of UAV

Acknowledgements

I would like to express my sincerest gratitude towards my advisor, Dr. David Boyle for all the guidance and support he has provided during my Master's thesis. The lessons I learnt from him have had a great impact on my thinking and the way I approach problems. I would also like to express my special thanks to my supervisor, Dr. Vesa Hirvisalo, for all the advice and feedback he has given me on my thesis. I also thank Prof. Vlado Handziski for his valuable guidance, inspiration and teachings during my time at TU Berlin that motivated me to work in this field and with UAVs and robots.

I would like to thank Imperial College London for providing an intellectually stimulating environment and a place where I could devote a tremendous amount of time for working on this thesis and receive feedback and guidance from colleagues. I am particularly grateful to Aalto University and TU Berlin. It is a real privilege to complete my graduate studies at such prestigious universities.

I would also like to thank Manasi for proof-reading my thesis and helping me shape the way it is. Last but not the least, I would also like to thank my family for their encouragement and constant support.

Espoo, 31.7.2017

Laksh Bhatia

Contents

Abstract	ii
Acknowledgements	iii
Contents	iv
Abbreviations and Node Details	xi
1 Introduction	1
1.1 Aim	1
1.2 Specific Contributions	1
1.3 Background and Related Work	1
1.4 Approach and Thesis Structure	2
2 Scenarios	4
2.1 Introduction	4
2.2 Wind-farms	4
2.3 Oil and Gas Pipelines	7
3 System Architecture, Hardware, Modelling and Simulation Tools	9
3.1 Introduction	9
3.2 System Architecture	9
3.3 Hardware Tools	10
3.3.1 Openmote Hardware	10
3.3.2 TelosB	12
3.3.3 Unmanned Aerial Vehicle	12
3.4 Software Tools	13
3.4.1 ContikiOS	13
3.4.2 Simulation	13
3.4.3 Modelling	14
4 Communication Protocol Design	15
4.1 Introduction	15
4.2 Protocol Requirements	15
4.3 Protocols Analysed	16
4.3.1 ContikiMAC	16
4.3.2 TSCH	17
4.3.3 Rime Communication Stack	17
4.4 Design Choices	19

4.5	Proposed Protocol for communication between UAV and FFD	20
4.6	Protocol Stack	23
4.7	Protocol-Specific Parameters	24
5	Simulation, Modelling, Parameter Tuning and Static Node Evaluation	25
5.1	Introduction	25
5.2	Simulations	25
5.2.1	MAC Comparison Test	25
5.2.2	ContikiMAC and Rime Tests	26
5.3	Parameter Tuning	29
5.3.1	Tests	29
5.3.2	Results and Analysis	30
5.4	Modelling	32
5.4.1	Power Consumption Modelling	32
5.4.1.1	Analytical MAC	33
5.4.1.2	Protocol Power profile	35
5.4.2	Speed versus Reliability Modelling	37
5.5	Protocols' Static Nodes Evaluation	38
5.5.1	Tests	38
5.5.2	Results and Analysis	39
6	Field Tests	43
6.1	Introduction	43
6.2	Protocols' Evaluation with Moving Node	43
6.2.1	Tests	43
6.2.2	Results and Analysis	44
6.3	Outdoor Deployment and Evaluation	47
6.3.1	Tests	47
6.3.2	Results and Analysis	49
6.4	Requirements Satisfied	52
6.5	Conclusion	53
6.6	Future Areas of Research	54
7	Conclusion	55
	Bibliography	57
	Appendix A UAV modifications	60
	Appendix B Protocol Diagram with Hidden UAV node	63
	Appendix C Mathematical Models	65
C.1	Current Consumption Model	65
C.2	Speed vs Reliability Model	66

Appendix D Methods to approach 100% reliable, robust and handling hidden terminal issues	67
D.1 Reliability Tests	67
D.2 Multiple Nodes/Hidden Terminal Tests	71
D.3 Robustness Test	73
Appendix E Resources	75
E.1 Code	75
E.2 Data	75

List of Figures

2.1	Wind-farm in the North Sea ¹	6
3.1	System Architecture	10
3.2	OpenMote Hardware Ecosystem (From L to R: Openmote-CC2538, OpenBattery, OpenBase, OpenUSB)	11
3.3	DJI Matrice 100	12
3.4	Cooja Simulator	14
4.1	ContikiMAC Unicast Packet with Acknowledgement	16
4.2	ContikiMAC Broadcast Packet	17
4.3	Rime Communication Stack	18
4.4	Protocol Description with one static node. Protocol description with two nodes is shown in Figure B.1	22
5.1	UAV (Node 10) moving and static node (Node 5)	26
5.2	Timeline output shows the communication between UAV and FFD in Unicast mode	27
5.3	Multiple Unicast packets from FFD (node 5) to UAV (node 10)	28
5.4	Timeline output shows the communication between UAV and FFD in Reliable Unicast mode	28
5.5	UAV used for testing	30
5.6	Results when varying Channel Check Rate	31
5.7	Results when varying Power Output	32
5.8	Analytical MAC in MATLAB	33
5.9	Analytical MAC on Openmote-CC2538 with 10 packets	34
5.10	Analytical MAC on Openmote-CC2538 with 100 packets	34
5.11	Comparison between analytical MAC results from MATLAB and measured from CC2538 using a resistor of 10Ω in series with Openmote- CC2538 with a voltmeter in parallel to the resistor	35
5.12	Current vs. time for a single transaction of the protocol	36
5.13	Current vs. time comparison of the protocol created in MATLAB and measured from the node	36
5.14	Theoretical Maximum Speed of UAV for 100% data reliability	38
5.15	Table Test with 15 nodes	39
6.1	Corridor Test	44

6.2	Comparison between results from CTP and protocol from thesis . . .	47
6.3	Outdoor tests conducted in the marked region with a radius of 100m centred at Lat:51.575036, Long:-0.185947	48
6.4	RSSI values when the node is held 1m high and is moved away and back from the sniffer	50
6.5	RSSI values when the UAV is flown towards the sniffer from 150m away at an altitude of 10m	50
6.6	Communication range (in m) around the centre of the field at altitudes of 1m and 10m. The results for 1m were the maximum possible and the results for 10m are maximum achievable due to lack of space on the field	51
6.7	RSSI values when the altitude of the UAV is varied from 0 to 10m . .	51
6.8	Results from field tests overlaid on top of the modelling results	52
A.1	IEEE 802.15.4 Channel 26 in 2.4GHz spectrum at 7dBm output power	61
A.2	DJI Config File	62
A.3	Region A - DJI Controller communicating in ISM band, Region B - DJI Controller communicating outside ISM Band i.e. modified channel	62
B.1	Protocol Diagram with two nodes	64

List of Tables

4.1	IEEE 802.15.4 channels used in the protocol	19
4.2	ContikiMAC parameters	23
4.3	Protocol Parameters	24
5.1	Notations and description used in Figure 5.2	27
5.2	Notations and description used in Figure 5.4	29
5.3	Parameters and Results of a realistic model	37
5.4	Parameters and Results for Table Test with fixed network time	40
5.5	Results from Table Test separated by Node ID with fixed network time	40
5.6	Parameters and Results for Table Test with fixed number of packets .	41
5.7	Results from Table Test separated by Node ID with fixed number of packets	41
5.8	Parameters and Results for Table Test with fixed number of packets .	42
6.1	Parameters and Results for Corridor test	45
6.2	Results for Corridor test separated by Node ID	45
6.3	Parameters and Results for Corridor test with 15 nodes	46
6.4	Parameters for CTP test	46
A.1	DJI Lightbridge Frequency Division	60
C.1	Current and timings used for modelling of Current consumption . . .	65
D.1	Parameters and Results for Corridor Test 1	68
D.2	Results for Corridor Test 1 by node ID	68
D.3	Parameters and Results for Corridor Test 2	69
D.4	Results for Corridor Test 2 by node ID	69
D.5	Parameters and Results for Corridor Test 3	70
D.6	Results for Corridor Test 3 by node ID	70
D.7	Parameters and Results for Corridor Test 4	71
D.8	Results for Corridor Test 4 by node ID	71
D.9	Parameters and Results for Table Test 1	72
D.10	Results for Table Test 1 by node ID	72
D.11	Parameters and Results for Table Test 2	73
D.12	Results for Table Test 2 by node ID	73
D.13	Parameters and Results for Table Test 3	74

D.14 Results for Table Test 3 by node ID 74

Abbreviations and Node Details

Abbreviations

CPS	Cyber-Physical Systems
IoT	Internet of Things
UAVs	Unmanned Ariel Vehicles
WSNs	Wireless Sensor Networks
WSAN	Wireless Sensor and Actuator Networks
FFD	Full Functioning Device (Node)
RFD	Reduced Function Device (Node)
TSCH	Time Synchronized Channel Hopping
MAC	Medium Access Control
MCU	Microcontroller
OS	Operating System
CCR	Channel Check Rate
RSSI	Received Signal Strength Indicator
LPM	Low Power Mode
OTA	Over-The-Air
dBm	Decibels per one milliWatt
m/s	meters/second
km	Kilometers
m	meters

Table Abbreviations

Name	Abbreviation
8Hz (UAV), 16HZ (FFD)	CCRHZ
11 (FFD), 25 and 26 (UAV and FFD)	Channels I
11 (FFD-Openmote), 25 and 26 (UAV and FFD)	Channels II

Node numbers

Lower Bits of MAC Address	Node Type	Logical ID number
236.182	Openmote	Node 1
159.127	Openmote	Node 2
233.148	TelosB	Node 3
114.136	TelosB	Node 4
107.131	TelosB	Node 5
159.123	Openmote	Node 6
62.13	TelsoB	Node 7
14.118	TelosB	Node 8
133.126	TelosB	Node 9

Chapter 1

Introduction

1.1 Aim

The thesis explores the merger between Unmanned Aerial Vehicles (UAVs) and Wireless Sensor Networks (WSN). The main aim of the thesis is to assess the possibilities of UAVs as mobile sinks for WSN deployment, and also to recharge nodes in WSN deployments.

1.2 Specific Contributions

This thesis is a Proof-of-Concept for using multi-rotor UAVs as data sinks with high reliability. The thesis describes a protocol for communication between the UAVs and nodes deployed on the ground. The protocol described is a novel asynchronous communication protocol. The main contribution of the thesis is that the protocol has been designed and shown to approach a 100% data reliability in real-world environments.

The thesis presents an energy model for the nodes. The modelling is used to guide the development of recharging strategies using inductive wireless power transfer for the long-term operation of such systems. A speed versus reliability model is created to calculate the maximum speed at which the UAVs could travel and the protocol would function reliably.

A series of simulations and table tests were conducted to verify the protocols' correctness and reliability in an indoor setting. A number of field tests were conducted using a UAV to verify performance of the protocol. The outdoor tests were used to understand the practical problems when developing synergy between UAVs and WSNs.

1.3 Background and Related Work

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to monitor physical or environmental conditions [1]. Sensor networks may have actuation systems and such systems are

called Wireless Sensor and Actuator Networks (WSAN).

WSNs have been used to gather data from remote and inaccessible areas for a long time. It is not always possible to have a reliable internet connection to collect data from remote and inaccessible places. In the past, humans were sent to collect this data. Another common method to collect this data was to connect the gateway nodes to the internet with Ethernet or WiFi. With advancement in technologies, some solutions involving ground robots were developed and are proposed in [2, 3, 4]. However, it might be dangerous to send a person or there may not be internet connectivity as the areas may be inaccessible, so it is not always feasible. UAVs are devices that can easily replace human or a ground robot interaction in these places.

UAVs are aircraft that do not have human pilots. They are commonly referred to as drones. In the last few decades, the use of UAVs has significantly increased. Military drones have been around since the 1980s. However, drones for commercial purposes and hobbyists started gaining popularity in the last decade. The major use case of personal and commercial UAVs have been aerial photography and videography. One of the major drone providers is DJI [5] which has been in operation since 2006. As of 2016, the revenue of the company is over 1.5 billion dollars. In 2017, Gartner published a report [6] stating that more than 3 million personal and commercial drones would be shipped in 2017. The UAV market is rapidly growing, and there is a chance for UAVs to reach a larger audience than just being used for video and photography purposes.

UAVs may be ideal data collection agents for distributed sensors [7]. The idea of using UAVs as data mules is not new. Many ideas have been proposed in literature [8, 9, 10, 11] but all these papers and articles are simulation based. An opportunity exists to overcome many limitations of WSN technologies by synergising them with recent advances in small aerial robotics. This thesis is thus motivated by exploring this synergy in a practical capacity.

1.4 Approach and Thesis Structure

Most problems and solutions related to WSNs deployment have been scenario dependent. Hence, a general solution is not feasible given the fact that every scenario has different parameters. These parameters include the number of sensors, data size, collection and reporting frequencies, energy requirements, topology, the range of communication and physical constraints in the placement and size of nodes among others. For this reason, two scenarios have been described in Chapter 2 where the solution presented in the thesis could be ideal. Chapter 3 describes the hardware and software tools used for the purpose of this thesis. Chapter 4 describes the protocol that is developed and the protocols' stack used for simulation and real-world deployment. Chapter 5 describes the simulation, modelling and the first set of in-

door experiments performed. Chapter 6 presents the real-world deployments and the field tests conducted. Chapter 6 also highlights the areas of potential future work. The conclusion of the thesis is presented in Chapter 7.

Chapter 2

Scenarios

2.1 Introduction

Most WSN protocols developed are scenario dependent. Every application has different requirements based on a number of parameters. As part of the thesis, two possible scenarios are considered. These scenarios including the sensors used, the collection frequency required, and a typical data packet size are described in the following sections.

2.2 Wind-farms

The first scenario analysed is monitoring a wind-farm. A wind-farm is a geographical area where there are a number of wind-mills that are used to produce electricity. These farms are then connected to the power grid, and the generated electricity is transported. Due to a rise in the energy generated from renewable energies, a number of wind-farms are being constructed throughout the world. The size of a wind-farm depends on its electricity generation capacity which is in direct relation to the number of wind-mills it has. The size of a wind-farm could vary from 100 acres to 10000 acres.

Sensors are used to monitor the health of the wind-mills. They are also used to measure deterioration of the wind-mills, damage to the blades and/or any electrical problems. These sensors are mostly analog in nature, and the size of data is upper bounded by the resolution of the analog-to-digital converter.

The following sensors are used in the monitoring for wind-mills: [12]

- Side Mount Level Switches
- Electro-optic Level Switches
- Pressure Switches
- Pressure Transducers

Most wind-farms are constructed in remote and inaccessible areas for an unobstructed and clear path for the wind to flow. The current strategy to fix a problem involves sending an engineer to these regions. Based on the severity of the problem, additional resources could be requested to repair the windmill. This increases the downtime of the system. Another high ongoing cost of wind-farms is routine monitoring. The current solution for routine monitoring involves sending engineers to inspect wind-mills and collect sensors data manually.

The need for solutions for routine monitoring has seen companies create their own devices based on 802.15.4/Zigbee [13]. These devices help to communicate and gather data faster from wind-mills. However, current solutions still require a working internet connection to transfer data to a cloud server. Since most locations are remote, it is not always possible to have a stable 3G/4G connection. Also, most of these networks currently have a star topology, which means the gateway devices need to be one hop away from the sensors on the wind-mills. This makes current solutions quite impractical as multiple gateways would need to be connected to a cloud server with individual internet connections or a wired backbone connecting all of them to the web. Traditional WSNs are also not ideal for long-term deployments due to reasons like depletion of batteries, and other hardware and software problems. Many of these problems could be effectively mitigated, as shown in this work, by using UAVs in place of traditional networks.

The wind-farm that this thesis examines is the wind-farm that is currently being constructed in the English Channel [14]. Based on the requirements of that wind-farm and to generalise the solution, a grid of $N \times M$ nodes is considered where N and M number of wind-mills along the horizontal and vertical axes. Figure 2.1 is a picture of wind-farm called Amrumbank West in the North Sea.



Figure 2.1: Wind-farm in the North Sea¹

The proposed solution for the following scenario is as follows:

1. A single node gathering all data for a windmill every 15 minutes.
2. The size of data would depend on the resolution of ADC. (We assume that there are five sensors connected to every node and every sensor has a 2Byte reading. This means the total size of the packet is 10 Bytes)
3. The average distance between two wind-mills is 750m [15]. To communicate data between two nodes, low-power long-range radios are needed for point-to-point communication
4. Clusters can be formed with nodes that are in 1-2 hop vicinity from each other. The size of clusters is limited to 2 hop maximum as that would put the cluster-heads at a maximum distance of 1.5km which should cause minimal overlapping between two cluster-heads.
5. Multiple clusters can be formed based on their geographical locations or their ETX values.

¹Source: <http://www.elp.com/articles/2015/05/e-on-to-build-uk-offshore-wind-farm.html>

6. UAV flies over this wind-farm once per day to gather the data. If clustering is adopted, the UAV would only need to fly over some of the nodes instead of all of them, which would increase the distance it can cover.

The nodes need to be integrated with the flight controller to enable efficient path-planning and speed variation in the system. An added functionality to the system would be the ability of the UAVs to take images of the wind-mills as and when required or requested based on the system design.

2.3 Oil and Gas Pipelines

The second scenario analysed is monitoring of oil and gas pipelines. Oil and Gas Pipelines are used to transport the natural gas and oil that has been extracted from the earth. These drilling plants are offshore or in remote areas. Long lengths of pipe are used to transport these resources from one location to the other. Oil and gas pipelines span for 100s of kms and monitoring of these pipelines is a major challenge for the oil and gas industries. A number of sensors need to be deployed along the length of these pipes to measure their structural health.

The following sensors are used in Oil and Gas pipeline monitoring: [16]

- Pressure Transducers
- Solid State Pressure Switch
- Reliable Moving Parts Pressure Switches
- Continuous Level Transmitters
- Multi-Point Level Sensors
- Ultrasonic Level Transmitters
- Magnetostrictive Level Sensors
- Intrinsically Safe Zener Barriers
- Intrinsically Safe Liquid Level Controls
- Optic Level Switches
- Side-mounted Level Switch
- Solenoid Valves
- Latching Valves

Similar to the first scenario, the current strategy to identify and fix any damage is to send an engineer along the length of the pipeline to collect sensor data and determine the point of trouble.

In the past two years, UAVs have been used to monitor the gas pipelines [17] and to collect the video of the pipelines. Since the cameras use quite a lot of power, it is not possible to cover a substantial distance. Using sensor network systems would help to increase the distance that can be covered with the same drone. Since it is collecting sensor readings at precise locations instead of a video feed, the data could be processed on the UAV, and necessary decisions could be made.

The proposed solution for the following scenario is as follows:

1. Wireless Sensor nodes are deployed along the length of the pipeline, or wireless nodes are attached to the sensors already deployed on the pipeline.
2. Data collection frequency and the size of data are assumed to be similar to the wind-farms scenario.
3. UAV flies over the Oil and Gas pipelines deployment and collects data from them.
4. With a clustering algorithm, the data can be collected at a central gateway. This would reduce the number of devices that the UAV would have to communicate with.

An added functionality for this scenario would be that UAV could collect images or video at locations that seem to have erratic values that can be achieved by this integration.

Chapter 3

System Architecture, Hardware, Modelling and Simulation Tools

3.1 Introduction

This chapter describes the overall system architecture and the tools used for modelling, simulation and the hardware that was used for testing.

3.2 System Architecture

The overall system architecture is described in this section. The system consists of two main components UAVs and wireless sensor nodes. Some of the properties of the components are mentioned below:

- UAV
 - Autonomous control.
 - Path planning algorithms.
 - Wireless node connected to the UAV for communication with ground nodes.
 - Variable speed and range to ensure all data is received reliably.
- Wireless sensor nodes
 - Deployed on the wind-mills or pipelines.
 - Connected to multiple sensors.
 - Powered by batteries.
 - Can be powered by renewable energy sources like solar or wind energy.
 - Can be powered by wireless power transfer.
 - Radio Duty Cycling and Low Power Modes (LPMs).

This thesis targets the communication between UAV and nodes. The first component is the device that gathers sensor readings and the second one is the data mule or the sink to collect the sensor data. Wireless sensor nodes periodically sample the sensors. The mobile sink for this system is a cheap multi-rotor UAV. The UAV used is not autonomous and is controlled with the help of its controller. The UAV is flown within communication range of the nodes, and it gathers data from the nodes. A pictorial representation of the system is shown in Figure 3.1.

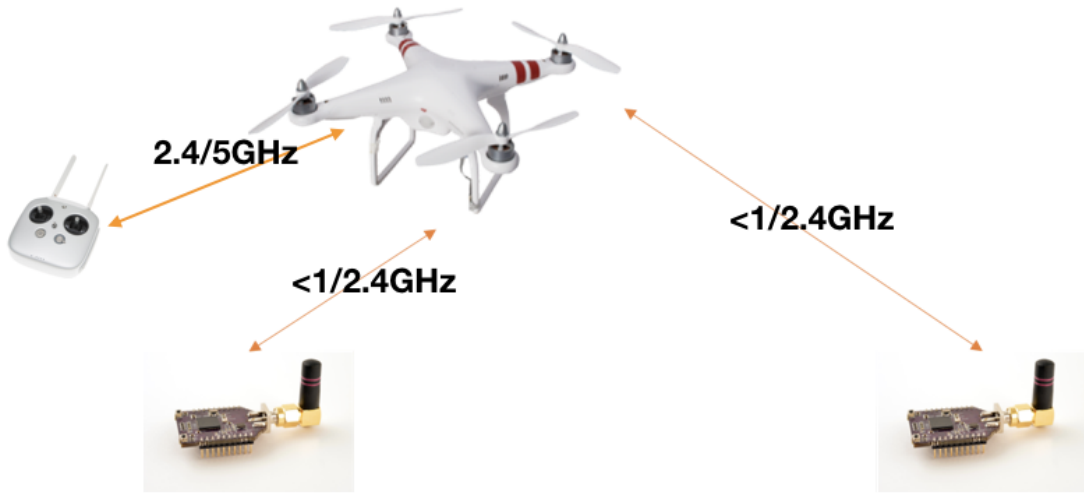


Figure 3.1: System Architecture

3.3 Hardware Tools

The hardware tools that were chosen for practical experimentation were the Openmote [18], TelosB [19] and the DJI Matrice 100 [20]. Openmote was chosen as it is an open source and a well-known wireless sensor node. It also has a larger memory size compared to TelosB nodes. TelosB was used where there were lower memory requirements. DJI Matrice 100 was selected as it is an open and highly customisable UAV platform. All the hardware used was bought commercially off-the-shelf. For commercial applications, scenario specific hardware can be designed.

3.3.1 Openmote Hardware

The Openmote hardware ecosystem consists of 4 major devices. These devices are pictured in Figure 3.2.

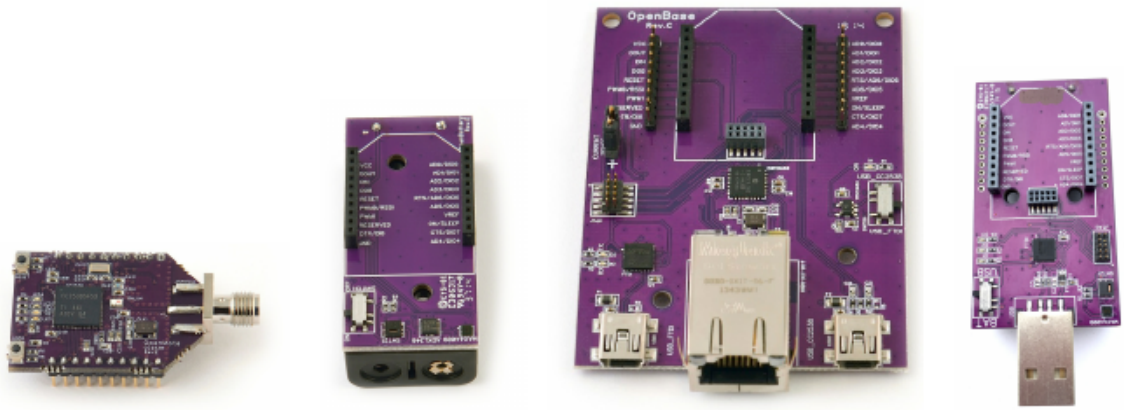


Figure 3.2: OpenMote Hardware Ecosystem (From L to R: Openmote-CC2538, OpenBattery, OpenBase, OpenUSB)

Openmote-CC2538 is the brain of the whole system. It consists of a Texas Instruments CC2538xFnn chip. CC2538 has a Cortex-M3 MCU with an IEEE 802.15.4 compliant radio transceiver. The clock speed can go up to 32MHz. It has a 256KB in-system-programmable flash which gives it support for Over-The-Air (OTA) updates. The radio communicates in the 2.4 GHz ISM band. It has a receiver sensitivity of -97dBm and a maximum output power of 7dBm. It has three low-power modes which enable the node to run for extended periods of time. The lowest power mode 2 consumes 0.4 μ A but requires an external trigger to wake-up.

OpenBattery is an add-on device for the Openmote-CC2538. It consists of a slot for the Openmote-CC2538, an on-off button, 2 AAA battery holder and three sensors: temperature, light and an accelerometer.

OpenBase is another add-on and consists of some useful chips and interfaces. It has a slot for Openmote-CC2538. It also has an FTDI chip that is used to convert USB to serial data. This chip can be used to upload code to the CC2538 and also to extract debug information from the CC2538. The OpenBase also has a 10-pin JTAG interface for debugging, and an Ethernet chip that can be used to connect CC2538 to the internet directly. There is also a power switch to switch between the USB and JTAG power. The CC2538 can be programmed via JTAG using Seggar JLink.

OpenUSB is another add-on that makes the Openmote-CC2538 similar to TelosB [19]. OpenUSB consists of the male USB connector, three sensors similar to OpenBattery, a JTAG connector and a slot for the Openmote-CC2538. Both JTAG and USB connectors can be used for debugging and flashing the CC2538.

3.3.2 TelosB

TelosB is an open source platform designed to enable experimentation for the research community. TelosB bundles tools for USB programming, CC2420 IEEE 802.15.4 radio with integrated F-type antenna, a low-power Texas Instruments MSP430F1611 MCU with extended memory. The nodes used are produced by Advanticsys based on the design created at UC Berkeley in 2004.

3.3.3 Unmanned Aerial Vehicle

A number of commercial drones are available in the market. However, for the purpose of practical experimentation, the DJI Matrice 100 was used. The UAV is shown in Figure 3.3. This is a research oriented UAV platform. Even though this UAV was used for the tests, a more careful selection would be needed based on the application requirements for other scenarios or realistic deployments.



Figure 3.3: DJI Matrice 100

DJI Matrice 100 is an open platform and is highly customisable. It is a stable and powerful flying platform. The platform has the N1 flight controller and DJI LightBridge [21], giving the UAV a range of 3.5 km (line of sight). It also has two slots for batteries. Each battery gives it a hovering time of 40 minutes without the Zenmuse X3 (gimbal). With the Zenmuse X3, the UAV has a hovering time of 20 minutes. The maximum speed of Matrice is 18m/s in the GPS mode. A simple mathematical calculation gives us that at top speed in autonomous mode, the UAV

could cover approximately 20 km distance with a single battery or 38sq.km. For most applications, this could be enough distance that would need to be covered in a single flight plan. DJI Inspire 2 is another UAV that could be considered. The maximum speed of this UAV is 94km/h (26m/s) with a flight time of 27 minutes. This UAV could cover a lot more distance compared to DJI Matrice 100. However, as Matrice 100 is an open platform, it makes it suitable for research.

3.4 Software Tools

To select the best operating system for the purpose of this thesis, a number of real-time operating systems were assessed. They were RIOT [22], TinyOS [23], ContikiOS [24] and FreeRTOS [25]. RIOT has an implementation for 6LowPAN and a few MAC layer protocols. ContikiOS has implementations of the ContikiMAC and TSCH. TinyOS has support for 6TiSCH. It was decided that ContikiOS would be a good choice for the thesis for many reasons. One of the reasons was that it had support for the chosen hardware. It also supported a number of the MAC layer protocols that were to be evaluated for the thesis. ContikiOS also has a well-established community support. ContikiMAC and TSCH have been further described in Sections 4.3.1 and 4.3.2. For the purpose of modelling and simulating the network, MATLAB and Cooja were chosen, being popular tools with community support.

3.4.1 ContikiOS

ContikiOS is a lightweight open source operating system (OS) built around an event-driven kernel written in C. It connects battery-operated and low-power objects to the Internet. It is a highly portable operating system. The first version of ContikiOS was released in 2003. Since then a lot of different protocols on all layers of the OSI stack [26] have been written for the OS. It has become one of the most popular operating systems for low-power embedded devices and has been used for research as well as commercial applications. The code for the protocol described in the thesis is written in C and compiled using GCC.

3.4.2 Simulation

Cooja is the network simulator provided by ContikiOS. Zolertia Z1, TelosB, MSP430-F5438 with CC2420 are some of the mote types that can be emulated at the hardware level. Cooja allows the users to inspect the behaviour of the network and check the correctness of a protocol. However, it cannot simulate Openmote-CC2538, and so the Zolertia Z1 nodes are used for the purpose of simulation. Simulations are done to ensure the reliability of the communication protocol developed before deploying them into the hardware. A typical Cooja session is shown in Figure 3.4.

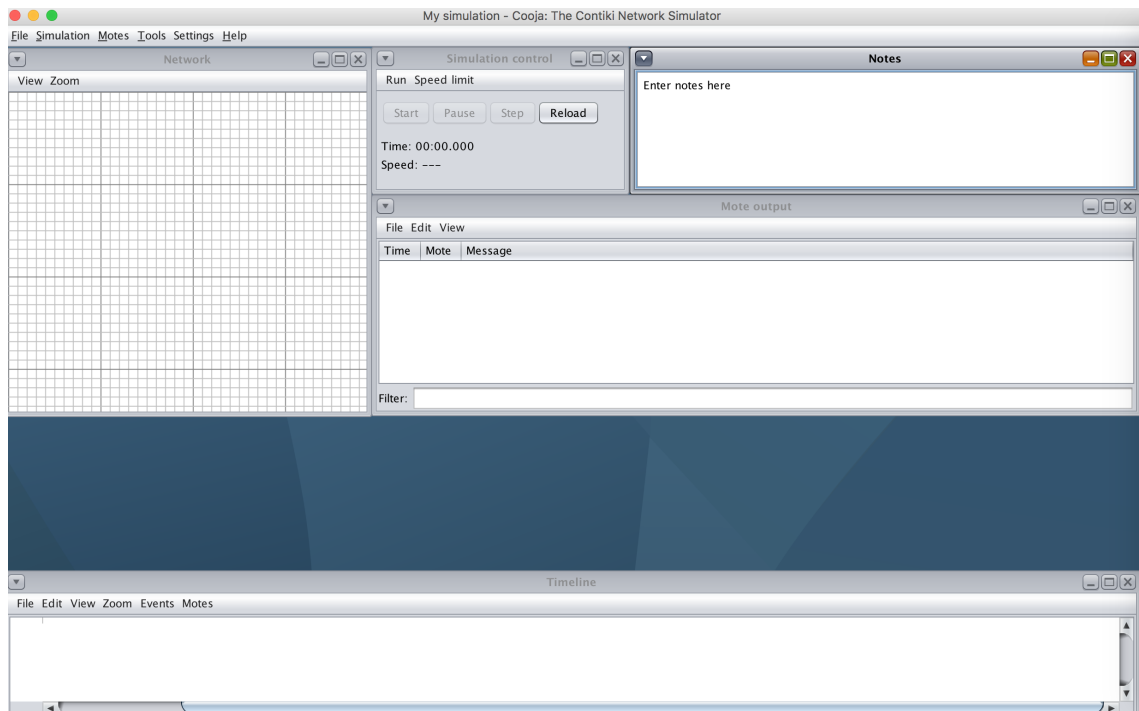


Figure 3.4: Cooja Simulator

For the purpose of simulating a UAV in Cooja, the mobility plugin [27] was used. This plugin enables an autonomous movement of the nodes specified by time and coordinate.

3.4.3 Modelling

For this thesis, MATLAB (matrix laboratory) was used for numerical analyses based on the mathematical models of the system's components of interest, e.g. Energy Model in Section 5.4.1.1. The numerical analyses were compared to practical experiments to ensure that the models were a close approximation of reality.

Chapter 4

Communication Protocol Design

4.1 Introduction

This chapter describes the requirements of the protocol, design choices and the protocols' implementation. It describes the MAC layer protocols considered for the communication protocols' design. This chapter also describes some of the problems faced and the modifications made to the protocol to mitigate those problems.

4.2 Protocol Requirements

The requirements for the the protocol are listed below. It should,

- Work with a mobile aerial sink: Enable UAV and static node communication.
- Have 100% data reliability: All the data from ground nodes should be received by the UAV.
- Data transfer needs to be quick so the UAV can cover maximum distance in minimum time.
- Work reliably in harsh conditions: E.g. for the offshore wind-farm scenario, the environment could be very humid and the actual communication range could be low. So, the packets need to be delivered to the UAV.
- Be robust and scalable.
- Be able to handle hidden terminal problems: the UAV would be travelling at high speed and overhearing nodes need to be dealt with.
- Be extensible to enable clustering: to ensure that the UAV can gather as much data as possible without compromising on speed, clustering algorithms need to be deployed so the number of nodes UAV would communicate with are low.
- Be energy efficient: E.g. protocol needs to be deployed on nodes on wind-mills that have an average lifetime of 25 years, so nodes need to be energy efficient.

4.3 Protocols Analysed

The physical layer parameters, including 2.4GHz ISM spectrum, maximum output power of 7dBm, receiver sensitivity of -97dBm are determined based on the hardware selected¹. The next layer of the OSI stack that has an impact on the protocol is the Medium Access Control (MAC) layer.

Synchronous and asynchronous MAC layers were analysed based on the implementations available in ContikiOS. ContikiOS has an implementation of TSCH and ContikiMAC. TSCH is a synchronous MAC protocol and ContikiMAC is an asynchronous MAC protocol. ContikiMAC, TSCH and Rime communication stack are described in the subsequent sections.

4.3.1 ContikiMAC

ContikiMAC is a radio duty cycling protocol. It uses periodical wake-ups to listen for packet transmissions from neighbours. If a packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. When the packet is successfully received, the receiver sends a link layer acknowledgement. To transmit a packet, a sender repeatedly sends its packet until it receives a link layer acknowledgement from the receiver. Packets that are sent as broadcasts do not result in link-layer acknowledgements. Instead, the sender repeatedly sends the packet during the full wake-up interval to ensure that all neighbours have received it. [28] The principle of ContikiMAC is shown in Figure 4.1 and Figure 4.2.

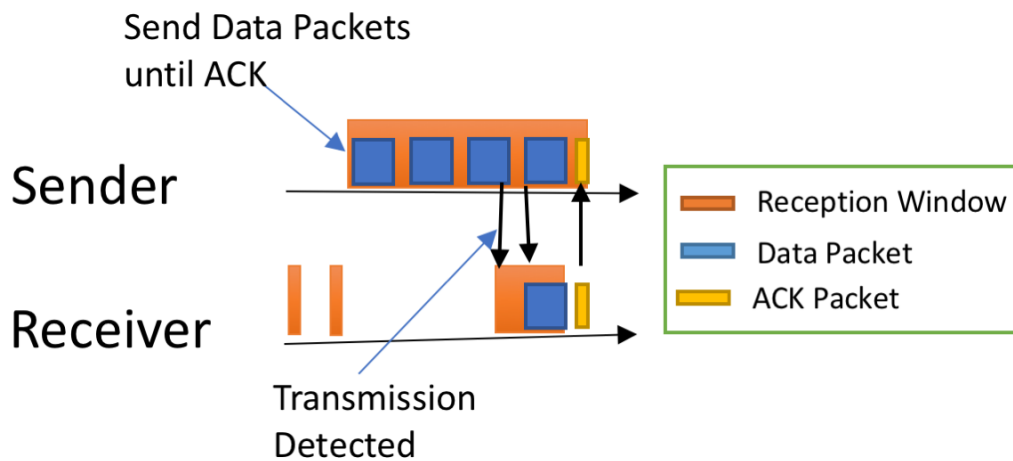


Figure 4.1: ContikiMAC Unicast Packet with Acknowledgement

¹Refer to Section 3.3.1 for other physical layer parameters

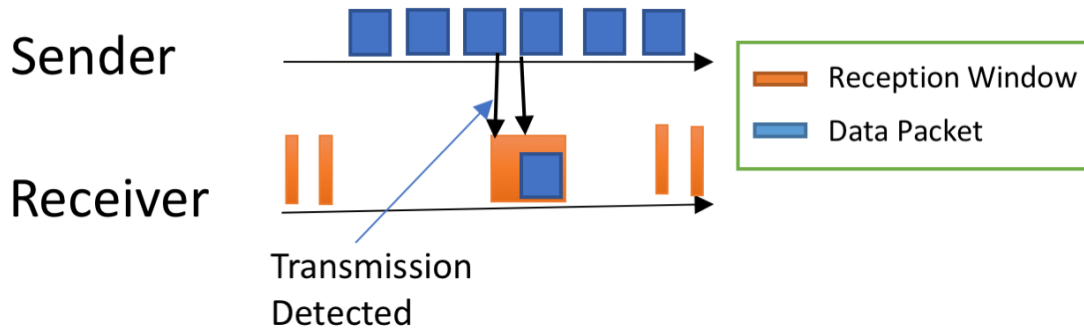


Figure 4.2: ContikiMAC Broadcast Packet

4.3.2 TSCH

Time Synchronised Channel Hopping (TSCH) [29] is a MAC protocol which was an amendment to the original 802.15.4 standard. It is standardised in IEEE 802.15.4e. As the name suggests, it is a time synchronised protocol. As part of this protocol, the network has a start-up period where the nodes in the network synchronise their clocks and decide on the schedule for channel hopping. This protocol is ideal for stationery networks, as maintaining up-to-date routes is too costly if some nodes are not static. For this reason, an extension to the TSCH is proposed in [30] which tries to introduce mobile nodes in the network. There is a set-up overhead when TSCH was used for this thesis. It was concluded that an asynchronous MAC is likely to be much effective.

4.3.3 Rime Communication Stack

The Rime communication stack provides a set of lightweight communication functions and implementations of best-effort, reliable and bulk sending and receiving. The protocols in Rime stack are layered which can be seen in Figure 4.3. The lower layers are less complex protocols, and as you go up the stack, the complexity of the protocol implemented increases.

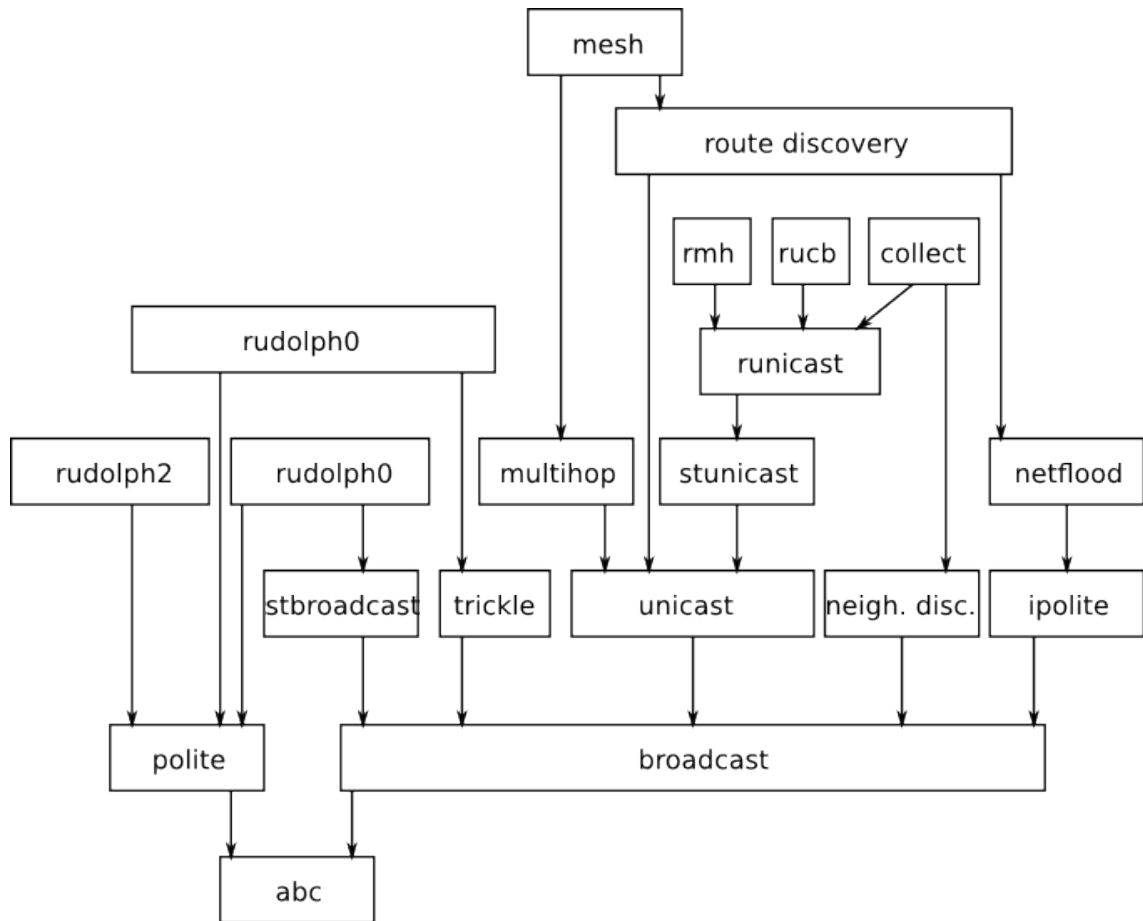


Figure 4.3: Rime Communication Stack

The communication protocols that were used from the rime stack for the purpose of this thesis are:

- broadcast: This protocol is the best effort local area broadcast i.e. packets are sent to all neighbours with a broadcast address, and no acknowledgement is expected.
- stbroadcast: This protocol is based on the broadcast protocol. At this layer, the node keeps broadcasting a packet after a fixed time.
- unicast: This protocol is based on top of the best effort local area broadcast. The packet is sent only once. Moreover, unlike the broadcast protocol, it expects an acknowledgement, and the address fields contain the address of the receiver.
- runicast: This protocol is the reliable unicast protocol. In this protocol, the address fields have the address of the receiver, it expects an acknowledgement, and if an acknowledgement is not received after a fixed time the packet is sent again.

4.4 Design Choices

This section highlights the design choices made for the protocol.

The first design consideration was to choose between a synchronous and an asynchronous MAC protocol. For this, experiments were conducted with the default TSCH and ContikiMAC examples in ContikiOS. Intuitively and later experimentally, it was realised that synchronous MAC would not be an ideal choice. The time it takes to form a schedule when the UAV is in the vicinity of a Full Functioning Device (FFD) is a couple of seconds. This would lead to a limit on the speed of the UAV. It was decided to use the more favourable asynchronous ContikiMAC implementation in ContikiOS.

The second design choice was to have multiple channels. Multiple channels help to overcome the problem of the hidden terminal as the actual data exchange takes place between the nodes on a different channel. Multiple channels also limit over-hearing and idle listening which improves energy efficiency of the nodes. Since three channels are used, a dual channel check was implemented for the FFD. FFD does alternate checks on the channels it has to communicate with the UAV and the Reduced Functioning Devices (RFDs). The IEEE 802.15.4 channels used for communication are described in Table 4.1. Even though the protocol designed in the thesis does not describe the communication between FFDs and RFDs, it has been designed to enable the possibility to effectively implement clustering algorithms to reduce the number of nodes a UAV would need to visit for data collection only.

Channel Number	Used for
Channel 11	Data exchange between RFD and FFD
Channel 25	Data exchange between UAV and FFD
Channel 26	Control messages between UAV and FFD

Table 4.1: IEEE 802.15.4 channels used in the protocol

The third decision was to use stubborn broadcast instead of sending broadcasts after fixed interval of time. This design choice helped to keep the main application layer free from being locked up by a process. Another reason it was used is because there were no energy constraints on the UAV.

The fourth design choice was to use reliable unicasts² in the transactions that were critical to the protocol. When the first set of field tests were conducted, a problem with the DJI controller and DJI lightbridge technology was discovered. This problem and its solution have been described in detail in Appendix A. Even though this problem was fixed, the DJI controller still had periodic communication in the 2.4GHz ISM band. So reliable unicast was chosen.

²Reliable unicast packets are sent until an acknowledgement is received

The final design choice was to use bulk acknowledgement when the actual data was sent to the UAV. The amount of data that needs to be sent from an FFD to UAV can be high. All the data that needs to be sent from the FFD are unicasted to the UAV one after the other. Once the data has been unicasted, the UAV bulk acknowledges the packets it received. Using a reliable unicast would add unnecessary time overhead to the protocol.

4.5 Proposed Protocol for communication between UAV and FFD

This section describes the protocol designed to meet the requirements mentioned in Section 4.2. The network has three major components. The first one is the UAV which is a moving node. The other two components are FFDs and RFDs.

The new messaging primitives used for protocol implementation are as follows:

- **REQUEST**: Request Data
- **REPLY**: Reply message to **REQUEST** data
- **ACCEPT**: Accept message
- **DATA**: Data packets
- **REQSEQ**: Request Sequence number
- **REPSEQ**: Reply to **REQSEQ** with specific sequence number
- **Backoff**: Sleep for a fixed time and drop **REQUEST** packets

FFD's and RFD's are deployed on the ground either on a wind-mill or along the gas pipeline based on the scenario. The protocol used for communication is as follows:

1. The network begins with the UAV starting a stubborn broadcast (4.3.3) of a **REQUEST** packet over a predetermined channel which is Channel 26.
2. The FFD is performing a Clear Channel Assessment (CCA) on two different channels, Channel 26 and Channel 11. Channel 11 is used for communicating with RFDs.
3. Once the FFD receives a **REQUEST** packet, it starts sending a reliable unicast (4.3.3) packet with a **REPLY** tag to the UAV.
4. On successful reception of the message from FFD, the UAV replies back with a reliable unicast packet with **ACCEPT** tag.
5. Once the **ACCEPT** packet is acknowledged, the UAV and FFD switch to Channel 25.

6. After a fixed time-out to ensure a successful channel switching, the FFD does a bulk unicast (4.3.3) of all the data it wished to send to the UAV.
7. Once the UAV has received a bulk of packets, it verifies whether or not any packets were dropped.
8. If packets were dropped, it does a reliable unicast of a packet with the **REQSEQ** tag. FFD replies with a reliable unicast of the requested packets with the **REPSEQ** tag.
9. Finally, the UAV does a reliable unicast of a packet with the **BACKOFF** tag and the back off time. After the successful reception of the packet or reaching the maximum number of retransmissions, the UAV switches back to Channel 26 and the protocol restarts.

The complete protocol described above is shown in Figure 4.4. Figure B.1 shows a scenario with two FFDs and one UAV. It describes how the protocol would handle the hidden node scenario.

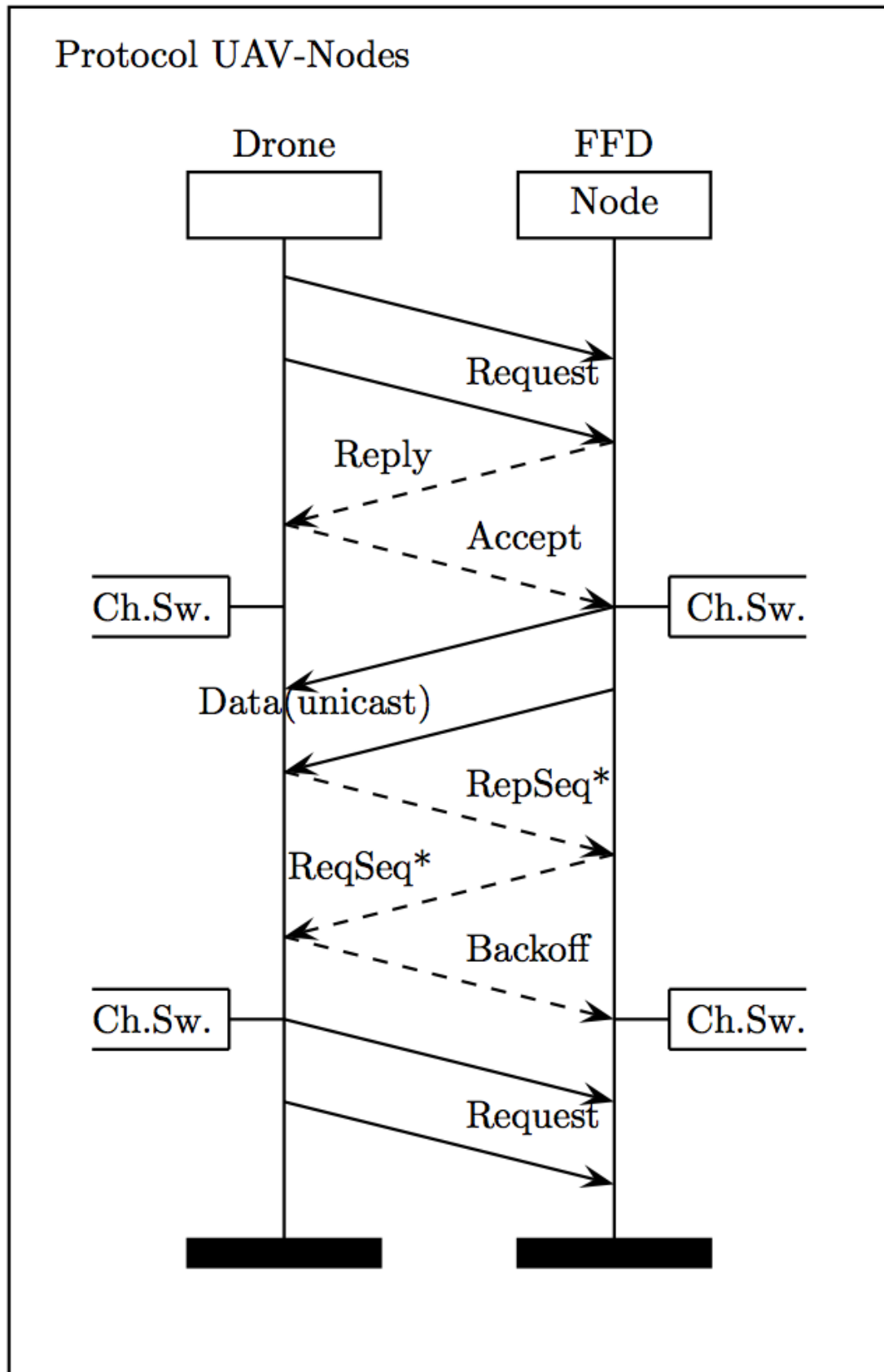


Figure 4.4: Protocol Description with one static node. Protocol description with two nodes is shown in Figure B.1

4.6 Protocol Stack

This section describes the protocols' stack. It describes the application level codes and the changes made to the underneath layers to implement the protocol. The protocol is implemented on top of ContikiMAC and Rime Communication Stack.³

The following parameters of ContikiMAC were modified

Variable Name	Value(s)	Description
Send Software Ack	1/0	Enable/Disable Software acknowledgements for received packets
Buffer Queue Size	20	Size of the Queue for max. number of packets at a time
Channel Change	1/0	Enable/Disable Channel Changing
Channel Check Rate	8/16Hz (125ms/62.5ms)	Channel check frequency
Max. Retransmissions	7/15/31	Maximum number of retransmissions for reliable unicast

Table 4.2: ContikiMAC parameters

Channel changing is an important part of the protocol. The FFDs perform CCA on alternate channels at a higher channel check rate such that the effective check rate for the UAV and the FFDs are the same. This change was performed in the file 'core/net/mac/contikimac/contikimac.c'.

For the Rime Stack, a new function was implemented for reliable unicast in the file 'core/net/rime/runicast.ch' to cancel a reliable unicast. The channel was noisy due to the problems mentioned in Appendix A, so there was a need to cancel a reliable unicast if the packet was successfully received, but an acknowledgement was not received. As the protocol sends packets from UAV to FFD and back in a sequential way, the packets can be considered as implicit acknowledgements.

A typical packet sent in any of the transmissions had an average of 40-byte length. A packet consisted of one of the messaging primitives⁴, sequence number of the current packet, the total number of packets that would be sent during the bulk transfer and the data that needs to be sent.

The Coffee File System [31] was also implemented to store records of every transaction that took place. A typical record consisted of a messaging primitive, sequence

³Refer to Appendix E for the codes.

⁴Primitives from Section 4.5

number, Received Signal Strength Indicator (RSSI), the number of retransmissions and the timestamp. After a transaction is complete, all messages are written to the flash memory of the CC2538.

4.7 Protocol-Specific Parameters

This section consists of the protocol specific parameters. The values of these parameters were varied for different tests, and adequate parameters were used for the tests conducted outdoors. The parameters are mentioned in Table 4.3.

Parameter Name	Parameter Description
Max Retransmissions	Maximum number of retransmissions for a reliable unicast.
UAV Retransmit Time	The stubborn broadcast time for the UAV (frequency at which REQUEST packets are sent).
Backoff Time	The backoff time for a FFD for which it would drop all packets received from the UAV or go into its lowest power state to conserve energy.
TX Power	The transmit power of nodes' Radio in dBm.

Table 4.3: Protocol Parameters

Chapter 5

Simulation, Modelling, Parameter Tuning and Static Node Evaluation

5.1 Introduction

This chapter describes the simulation work carried out in designing the protocol. Since there were many variables in the implementation of the protocol, the parameter tuning tests were performed to verify the impact of these parameters. The tests and their results are described in this chapter. Based on the parameters chosen, power modelling was carried out to approximate the lifetime of the device. This test gave an approximate idea about the lifetime of the network with a battery of fixed size and no possibility of recharging. It also guides the design of appropriate renewable system designs. This chapter also describes indoor/table tests conducted and their results. They were used to further tune the parameters prior to outdoor tests.

5.2 Simulations

The first tests were performed in simulation. These tests were carried out on Cooja Simulator. Each subsection describes the test, deployment scenario and the analysis of the results obtained.

5.2.1 MAC Comparison Test

The first test was a comparative analysis of synchronous and asynchronous MAC layers. TSCH was the synchronous MAC and ContikiMAC was the asynchronous MAC compared. Figure 5.1 shows the initial scenario with only two nodes. Node 10 is the UAV which is moving, whereas Node 5 is the static node. For both tests, the default examples of TSCH and ContikiMAC were flashed on the nodes.

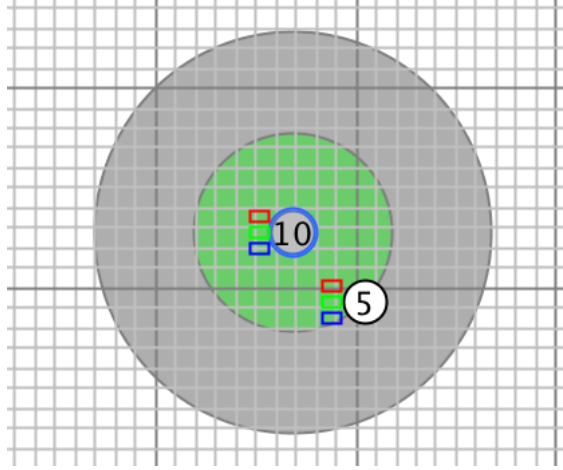


Figure 5.1: UAV (Node 10) moving and static node (Node 5)

The tests showed that TSCH has an initial schedule set-up time and when the UAV is moving, this set-up time was an overhead on the communication. If the speed of the UAV was high, the schedule was never set-up and no actual data was exchanged. If TSCH is used, the speed of the UAV would be bounded by the set-up time. The results of ContikiMAC were that the data exchange took place as soon as the UAV sent a broadcast packet when it was in the communication range of the FFD.

This test concluded that an asynchronous MAC is a better design choice for this protocol compared to a synchronous MAC. So, the idea of a synchronous MAC was dropped, and asynchronous MAC was used for subsequent tests. The asynchronous MAC used was ContikiMAC.

5.2.2 ContikiMAC and Rime Tests

A number of simulations were performed on Cooja to understand the working of the different primitives of the Rime Stack and how ContikiMAC sends these packets. They were then used to choose the right parameters for different parts of the protocol. A similar set-up to the one mentioned in the previous test was used.

The timeline outputs when the UAV or FFD sends unicast or reliable unicast packet were compared. Figure 5.2 shows the timeline output when the FFD sends multiple unicast messages to the UAV. Table 5.1 describes the notations used in the figure. Figure 5.3 shows the radio messages output when multiple unicast messages are sent by the FFD to the UAV.

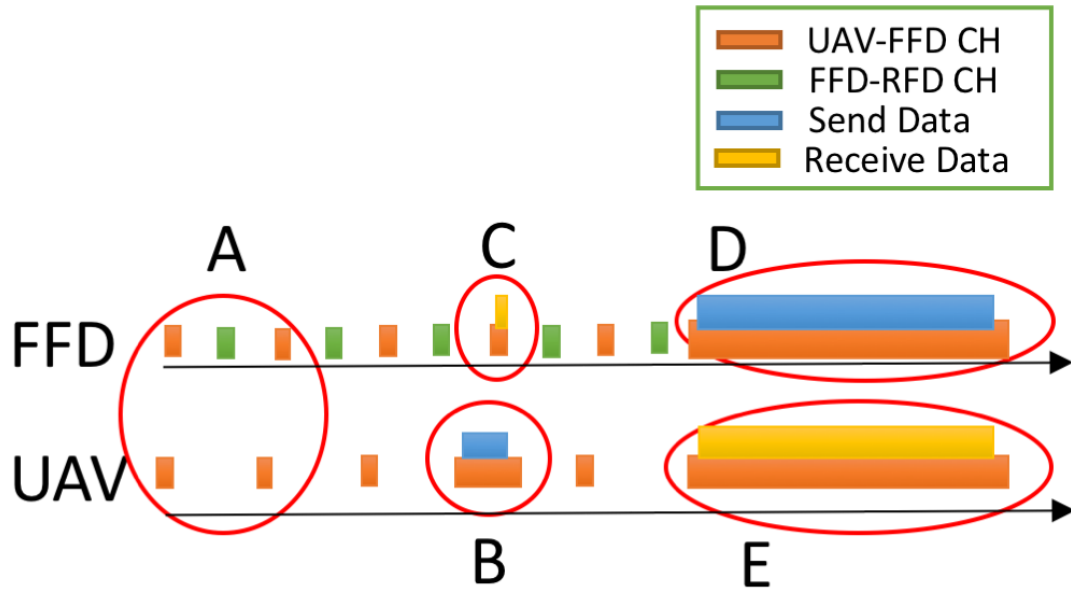


Figure 5.2: Timeline output shows the communication between UAV and FFD in Unicast mode

Notations	Description
A	8Hz check rate for UAV, FFD has a check rate of 16Hz
B	UAV sends broadcast message to FFD
C	FFD receives the broadcast message
D	FFD sends in unicast multiple packets to FFD
E	UAV receives the unicast packets

Table 5.1: Notations and description used in Figure 5.2

Radio messages: showing 19/922 packets				
File Edit Analyzer View				
No.	Time	From	To	Data
587+41	00:25.547	5	10	43: 0x519866CD ABFFFF00 05000E...
629+41	00:25.688	5	10	43: 0x519867CD ABFFFF00 05000E...
671+41	00:25.828	5	10	43: 0x519868CD ABFFFF00 05000E...
713+41	00:25.969	5	10	43: 0x519869CD ABFFFF00 05000E...
755+41	00:26.110	5	10	43: 0x51986ACD ABFFFF00 05000E...
797+41	00:26.251	5	10	43: 0x51986BCD ABFFFF00 05000E...
839+41	00:26.391	5	10	43: 0x51986CCD ABFFFF00 05000E...
881	00:26.533	5	10	43: 0x41986DCD ABFFFF00 05000E...

Figure 5.3: Multiple Unicast packets from FFD (node 5) to UAV (node 10)

Figure 5.4 shows the timeline output when the FFD sends a single reliable unicast message to the FFD. Table 5.2 describes the notations used in the figure.

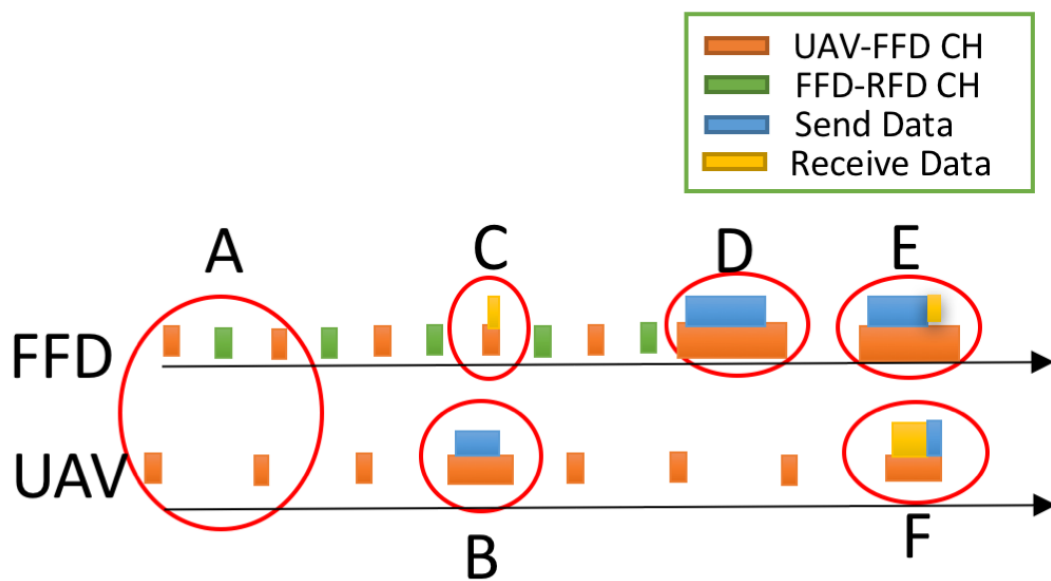


Figure 5.4: Timeline output shows the communication between UAV and FFD in Reliable Unicast mode

Notations	Description
A	8Hz check rate for UAV, FFD has a check rate of 16Hz
B	UAV sends broadcast message to FFD
C	FFD receives the broadcast message
D	FFD sends the packet, and the UAV does not receive it
E	FFD sends the packet again and waits for acknowledgement
F	UAV receives the reliable unicast and acknowledges it

Table 5.2: Notations and description used in Figure 5.4

Since the developed protocol has to be deployed on a fast moving sink, the chances of a single unicast packet reaching another node is low. This was verified with the initial set of field tests performed with the UAV and the node. So as to ensure that the packets were delivered reliably, reliable unicast was used when sending REPLY, ACCEPT, REQSEQ, REPSEQ and BACKOFF¹ packets. However, while sending DATA packets, unicast was used. These design decision were backed by Figures 5.2 and 5.4.

5.3 Parameter Tuning

5.3.1 Tests

After designing the protocol and deciding on the Rime stack primitives, the first hardware test was conducted to tune the parameters of the protocol. The test was done using two Openmote-CC2538 nodes. Both the nodes were flashed with the firmware using the OpenBase, and then one of them was put on an OpenBattery. The node with OpenBattery acted as the FFD, and the node on the OpenBase was the UAV-node. The debug output from the UAV-node was used to generate the results. Figure 5.5 shows the UAV-node which was kept under the UAV. The UAV and its controller were actively communicating outside the ISM Band when these tests were conducted².

¹Primitives described in Section 4.5

²Figure A.3 shows the controller communication outside the ISM band



Figure 5.5: UAV used for testing

As part of this test, Channel Check Rate (CCR) and transmit power was varied. The distance between the UAV-node and the FFD was fixed at 1m. Received Signal Strength Indicator (RSSI) and number of retransmissions were recorded as part of the test.

5.3.2 Results and Analysis

The data for the above tests was collected via the debug output and then passed through a script and plotted. The left y-axis of the plots are RSSI values in dBm, and the right y-axis is the number of retransmissions. The graphs show the mean for each of the parameters. Variations for each of those parameters have also been plotted.

In Figure 5.6 the CCR of ContikiMAC was varied from 4Hz to 16Hz in powers of 2, keeping the output power fixed at 7dBm, and 2 data packets were sent. The plot shows that with a CCR of 4Hz, the average number of retransmissions is much higher. The average number of retransmissions with CCRs of 8Hz and 16Hz were similar. However, the minimum number of retransmissions was 1 for 8Hz check rate, and there were no retransmissions when CCR was 16Hz. So, CCR of 8Hz and 16Hz were chosen for the protocol. For the next test, the CCR was fixed at 16Hz since it had no retransmissions.

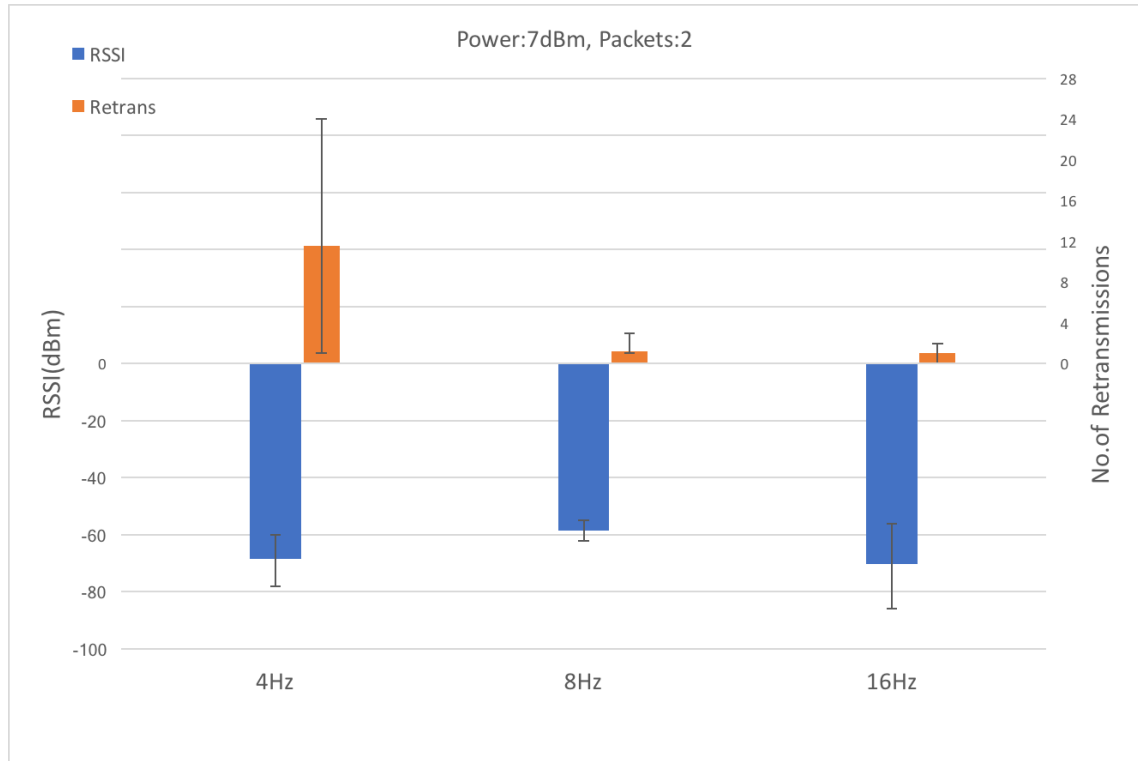


Figure 5.6: Results when varying Channel Check Rate

After fixing the CCR at 16Hz, the next parameter that was tuned was the transmit power of the nodes. RSSI values and number of retransmissions for 16Hz and 2 data packets are plotted in Figure 5.7. It is visible that the number of retransmissions with a transmit power of -24dBm was quite high. The maximum retransmissions for -24dBm was 20 although it never timed out. All other transmit power levels have similar plots.

The test concluded that transmit power by keeping the nodes at a fixed distance had no real impact on the number of retransmissions. However, the RSSI values recorded were used as a proxy for actual distance in field tests. In WSN, the actual distance for point-to-point communication is less significant than the communication distance between two nodes. So, the results of RSSI are used to calculate the distance of the UAV and FFD in the latter tests.

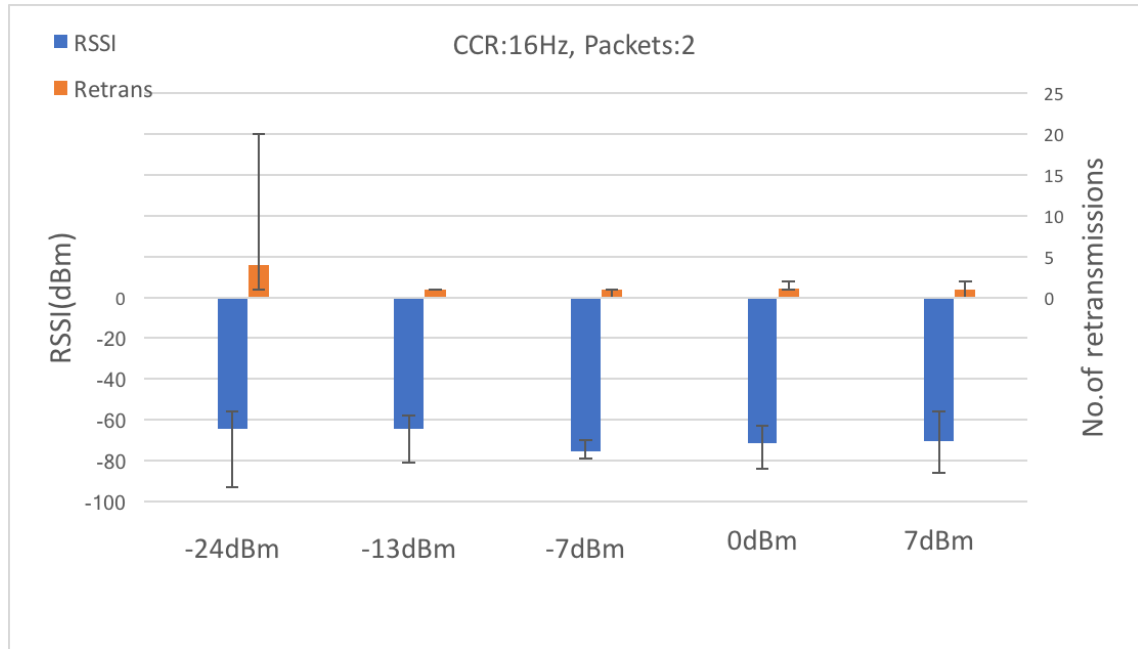


Figure 5.7: Results when varying Power Output

5.4 Modelling

5.4.1 Power Consumption Modelling

An important aspect of research work in sensor networks is to maximise the deployments' lifetime. To increase the lifetime of the network it is important to understand how the power is consumed by the nodes. So, a mathematical model was designed and an analysis was conducted using MATLAB to calculate the approximate power consumption by this protocol. The mathematical model has been described in Appendix C.1.

Determining the accuracy of results obtained from the developed model was difficult because of the complexity of the protocol. For this purpose, an analytical model was created. The simulation of this analytical MAC helped to determine a baseline performance which was then compared to the results obtained by running the same MAC on CC2538. The analytical MAC model simulated on MATLAB is described in Section 5.4.1.1. Section 5.4.1.2 describes the current consumption by the protocol and also the approximate lifetime of a node.

5.4.1.1 Analytical MAC

A micro-benchmark was done by implementing the same MAC layer in MATLAB and ContikiOS. The analytical model for the same is as follows:

- Turn on node (with or without radio)
- Node in Receive State
- Switch from Receive to Transmit State
- Send a packet
- Switch back from Transmit to Receive State
- Turn radio off

The timing and the current parameters for the model were taken from the datasheet of CC2538 [32]. The only user-defined parameter in this model was the amount of time used to take a sensor reading, create a packet and send it. This parameter was fixed at 5 seconds.

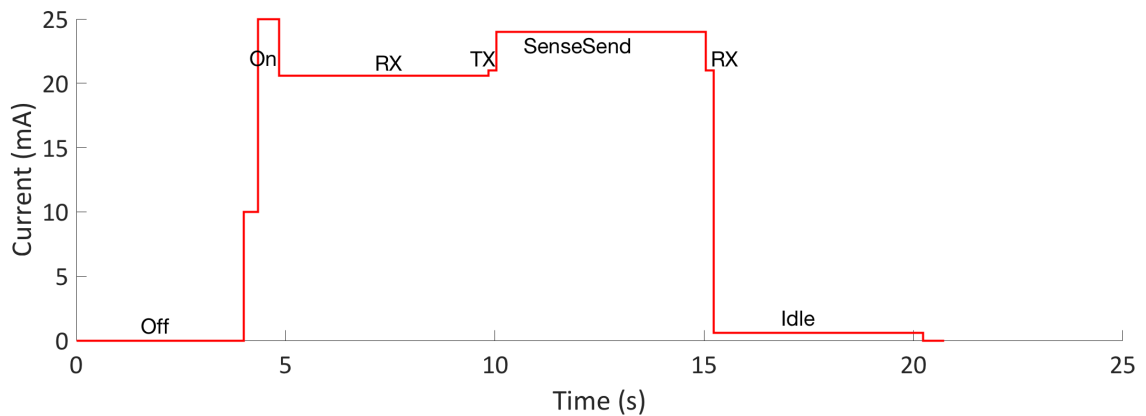


Figure 5.8: Analytical MAC in MATLAB

Figure 5.9 shows the current v/s time for the CC2538 running the analytical MAC protocol. For this test, the Radio was turned on with the MCU, and 10 packets each of 3Bytes were broadcasted.

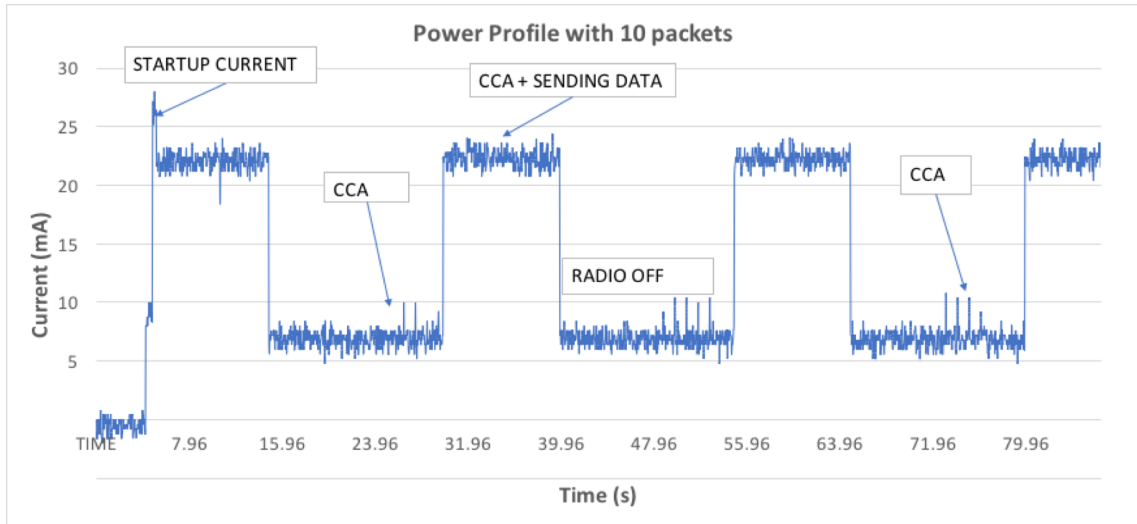


Figure 5.9: Analytical MAC on Openmote-CC2538 with 10 packets

Figure 5.10 shows the current v/s time for the CC2538 running the analytical MAC protocol. For this test, the Radio was turned on after the MCU booted and 100 packets each of 3Bytes were broadcasted.

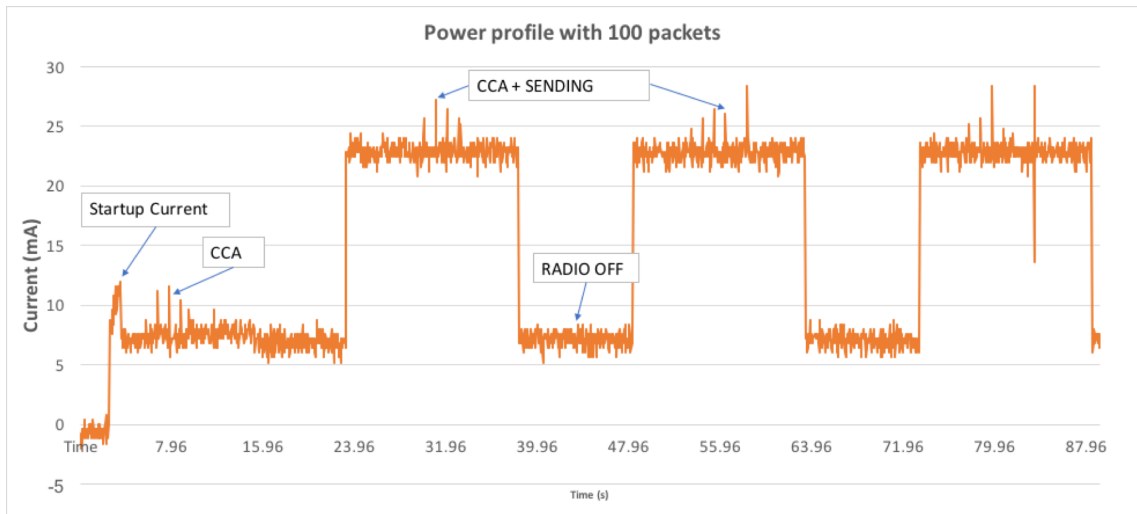


Figure 5.10: Analytical MAC on Openmote-CC2538 with 100 packets

The final Figure 5.11 shows the comparison between the results obtained from MATLAB and real-world implementation. This shows that the assumptions made while writing the model were fairly accurate and an approximation of the real situation. However, there is a significant difference in the IDLE mode. This difference is because the MATLAB code assumes the node goes into LPM 2 when in IDLE state where it consumes 0.6 μ A. However, when the code was run on the node, it

was drawing 6mA current due to the LEDS that were on in the IDLE state.

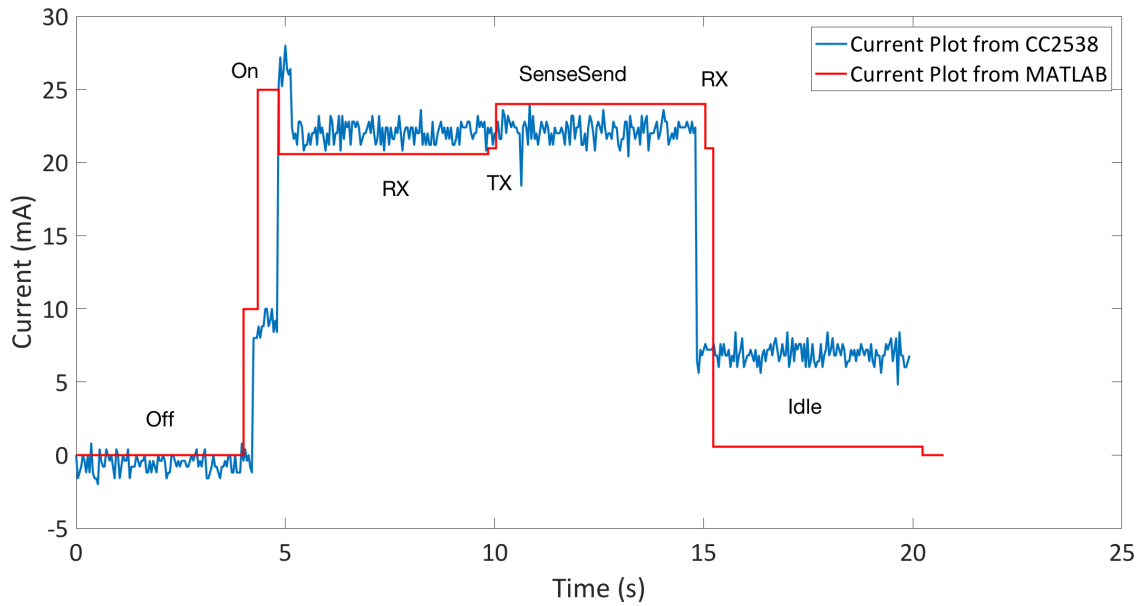


Figure 5.11: Comparison between analytical MAC results from MATLAB and measured from CC2538 using a resistor of 10Ω in series with Openmote-CC2538 with a voltmeter in parallel to the resistor

5.4.1.2 Protocol Power profile

Once the analytical MAC was verified, it was concluded that the mathematical model developed is sufficiently accurate. So, the parameters of the protocol were given as inputs to the model. The protocol was also flashed on to a CC2538 and the current was measured from it using a resistor of 10Ω in series with Openmote-CC2538 with a voltmeter in parallel to the resistor.

Figure 5.12 shows the current consumed by various operations in the protocol for one run of the protocol. Figure 5.13 overlays the plots from MATLAB model and from the node.

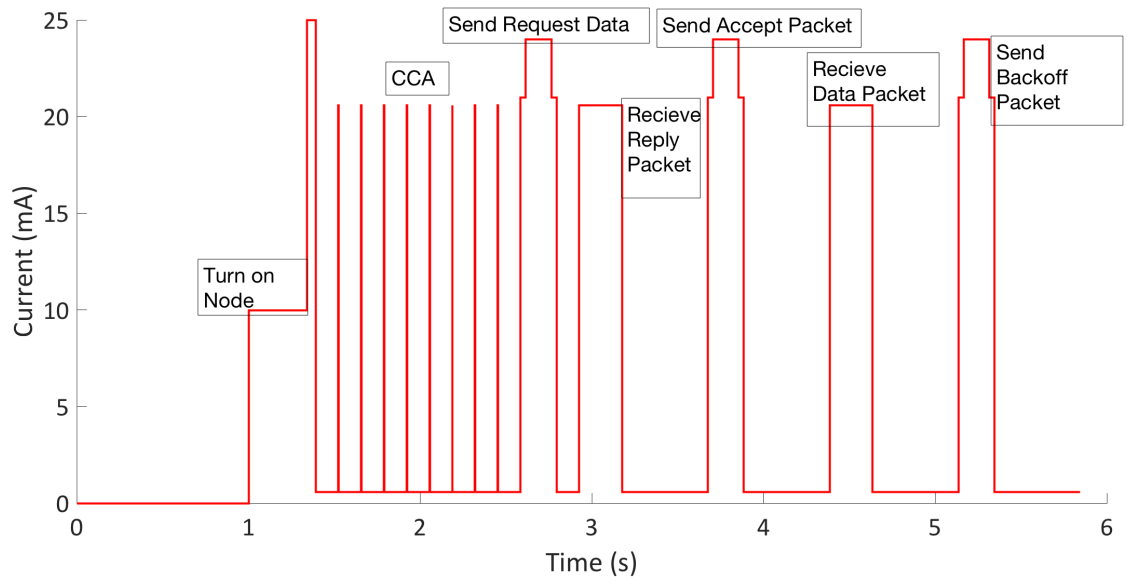


Figure 5.12: Current vs. time for a single transaction of the protocol

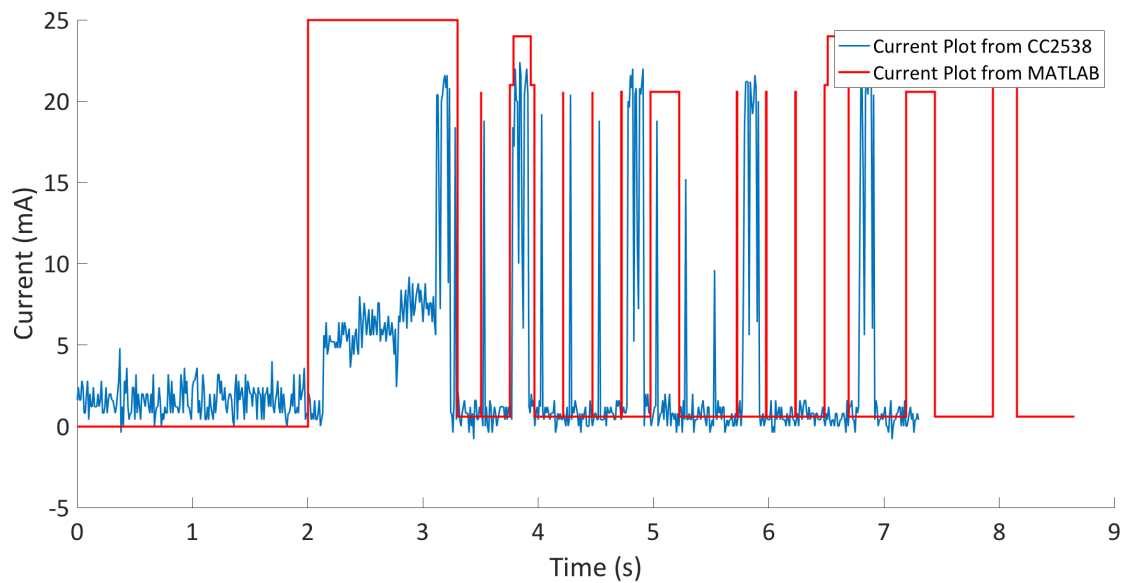


Figure 5.13: Current vs. time comparison of the protocol created in MATLAB and measured from the node

Based on the mathematical model developed, the average current required for the protocol is 6.14 mA, and the current consumed in 8 seconds is 27.35 mA for one transaction without accounting for retransmissions. The frequency at which the FFDs gather data is 15 minutes, and the communication with the UAV happens every hour. So the average current drawn for 1 hour with the node being at LPM 2

when there is no communication is 28.79 mA. A standard AAA battery is rated at 1000 mAH whereas an AA battery is rated at 2400 mAH. That means, the FFDs would last for 1133 days without recharging.

One of the major energy consumers for sensor networks is the sampling the sensors at regular intervals. This power drawn has been ignored in the above calculation. To make the model more realistic, the parameters for sensors were introduced in the model. They are described in Table 5.3 along with the expected lifetime of a node.

Parameter Name	Value
Number of sensors	5
Maximum current drawn per sensor	4.5 mA
Current draw for communication without retransmissions	27 mA
Size of battery	2400 mAh
Sleep Current	1.4 μ A
Total Current Draw	53 mA for 10 seconds
Average Current Draw	0.14 mA for 1 hour
Total Lifetime	672 days/ 1.8 years

Table 5.3: Parameters and Results of a realistic model

The table above shows that the lifetime of a single sensor node would be 1.8 years when no recharging or energy harvesting methods are adopted. The lifetime of a typical windmill is expected to be around 20-25 years. This warrants enough reason to make the system self-sustainable and have energy harvesting or recharging capabilities, so that the lifetime of the sensor nodes approaches that of a windmill.

5.4.2 Speed versus Reliability Modelling

A mathematical model was also created to understand the relationship between the speed of the UAV and the chances of achieving a 100% data reliability. The mathematical model has been described in Appendix C.2.

Figure 5.14 shows the plot of varying the speed from 0 to 50m/s and keeping the maximum communication range possible to 100m. The parameter hit distance decides when the transaction was initiated. In the graph, it is fixed at 0, meaning the transaction is initiated when the UAV enters the FFD's communication range. The minimum backoff time for every node was plotted on the y-axis. The minimum time required to complete one transaction with one data packet is approximately 3 seconds³. This time could be considered as relatively high, however, it is needed for 100% data reliability

³Refer to Figure 5.12 to see why it takes 3 seconds

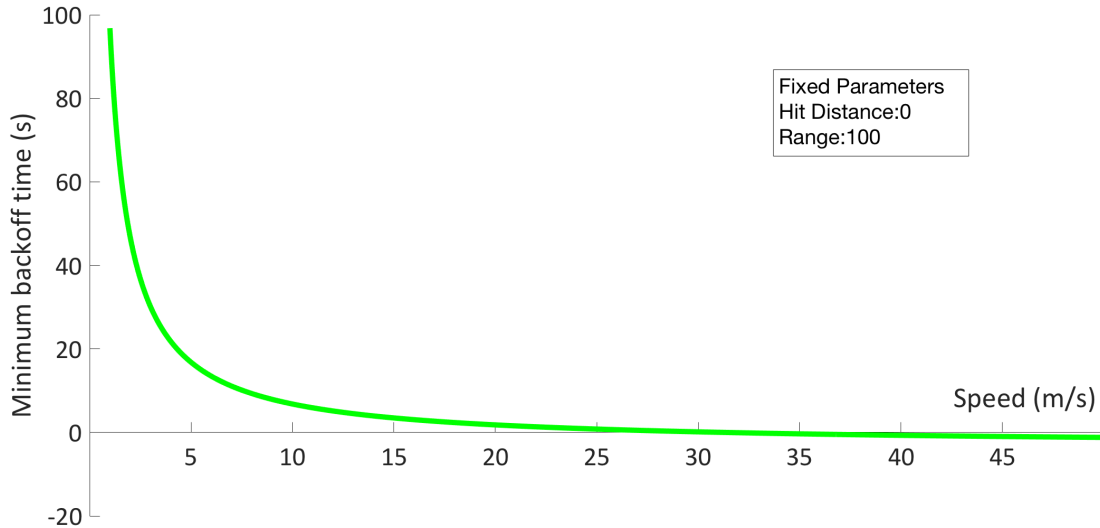


Figure 5.14: Theoretical Maximum Speed of UAV for 100% data reliability

The chances of data not being delivered was low when the free time approached zero. This was because the broadcast packet would reach as soon as the UAV was in communication range with the FFD. Also, it would have to stay in the communication range for at least 3 seconds. It was concluded that as the speed approaches 25m/s the backoff time tends towards zero. The transactions would not complete above that speed and a 100% data reliability cannot be achieved.

5.5 Protocols' Static Nodes Evaluation

5.5.1 Tests

The second hardware test was conducted with multiple FFDs with overlapping radio regions. The main aim of the test was to calculate the network time required to achieve a 100% data reliability. These tests test the scalability and robustness with respect to node density. The test was also used to calculate how many packets were received if the network time was constrained. This test had three parts which are described in the subsequent paragraphs. The following parameters were recorded by the UAV-node in all three parts of the test: RSSI, time, the sequence number of packet, the number of retransmissions, messaging primitive⁴ and the transmitter/receiver of the packet.

In the first part of the test, a maximum of 5 FFDs beside one UAV-node were in the network. All these devices were in overhearing range of each other. This helped test the robustness of the protocol. In this test, all nodes were stationary. This test was performed multiple times with different node configurations. Some of the tests

⁴Primitives from Section 4.5

had 2 Openmote-CC2538 and 3 TelosB nodes and others had 3 Openmote-CC2538 and 2 TelosB nodes. In this test, the network time was fixed, and a total number of packets received were recorded.

The second test conducted was similar to the first one. However, in this test, the number of packets were fixed, and the network was allowed to run until all the packets were received.

The third test conducted calculated the network time with 15 nodes in the network, each trying to send one packet to the UAV-node. 12 out of those 15 nodes were TelosB nodes, and the other 3 were Openmote-CC2538. These nodes are shown in Figure 5.15. It was performed to test the scalability of the network.

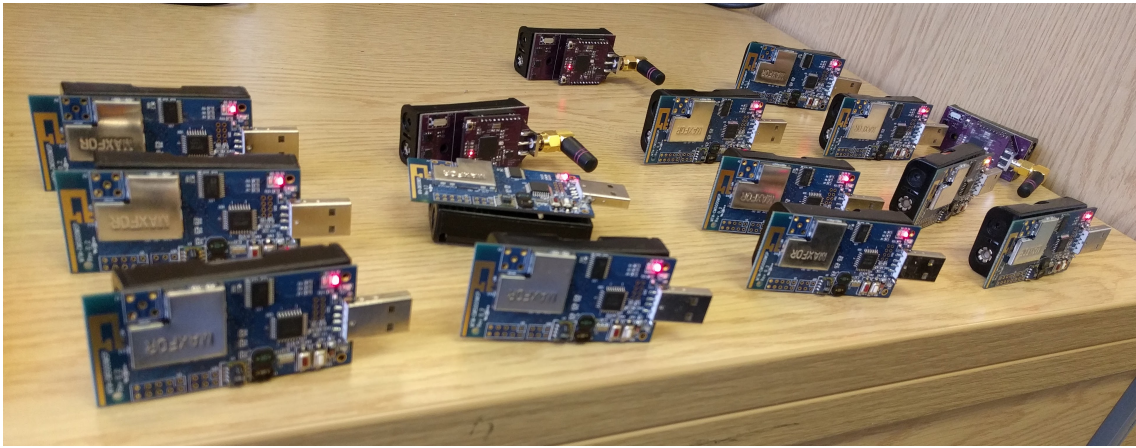


Figure 5.15: Table Test with 15 nodes

5.5.2 Results and Analysis

The results presented in this and Section 6.2 have two tables for every test performed. The first table lists the parameters and the main results of the test. The second table presents the results segregated by the node id's. This helped understand how the network worked at the node level instead of looking at the overall network performance. The second table presents the total number of packets received by every node, number of successful transactions by the node, success ratio and the churn. Success ratio is defined as the number of transactions completed divided by the total number of transactions expected. Churn is defined as the ratio of wasted transactions to total transactions completed, where wasted transactions is the ratio of the number of wasted packets to number of packets required to complete every transaction.

A series of results have been described in Appendix D.2 before the below results were obtained. The parameters for the first test along with the results are presented

in Table 5.4. Table 5.5 shows the results segregated by node ID.

Parameter Name	Value
Network Runtime	15 minutes
Power Output	-15dBm
Backoff time	30 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels II
No. of Nodes	5
Packets per node	30
Total number of Transactions	130
Total number of retransmissions	283
Total number of packets	529
Maximum Retrans achieved	3

Table 5.4: Parameters and Results for Table Test with fixed network time

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	110	27	90%	0.019
Node 2	104	26	86.67%	0
Node 6	106	26	86.67%	0.019
Node 8	96	24	80%	0
Node 9	113	27	90%	0.046
Total	529	130		0.017

Table 5.5: Results from Table Test separated by Node ID with fixed network time

The results from the above test showed that a 100% reliability could not be achieved. Due to retransmissions, 30 packets could not be received from each node in a 15 minute network time by the UAV-node.

For this purpose, the second test was conducted. In the second test, every node was required to send 30 packets and the network time was recorded as mentioned above. The results of the test are presented in Table 5.6 and Table 5.7.

Parameter Name	Value
Power Output	0dBm
Backoff time	30 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels I
No. of Nodes	5
Packets per node	30
Total number of Transactions	150
Total number of retransmissions	326
Total number of packets	610
Maximum Retrans achieved	3
Network Time	17 minutes

Table 5.6: Parameters and Results for Table Test with fixed number of packets

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	123	30	100%	0.025
Node 2	120	30	100%	0
Node 6	122	30	100%	0.017
Node 8	120	30	100%	0
Node 9	125	30	100%	0.042
Total	610	150		0.017

Table 5.7: Results from Table Test separated by Node ID with fixed number of packets

Table 5.7 shows that a 100% reliability was achieved when the network was kept running for 2 minutes more than the previous test. This test showed that the protocol designed in this thesis has 100% data reliability even when there are 5 nodes that can overhear each other. Another result that was presented from the logs from the UAV-node was that if at 7th retransmission the packet was not delivered, it wasn't delivered after that. So, the maximum number of retransmissions was reduced to 7. Reducing the number of retransmissions increased the overall efficiency of the system and reduced the energy wasted with the additional retransmissions.

The third test was conducted to check the scalability of the protocol. This test had 15 nodes in the network. Table 5.8 shows the parameters for the test and the results. A table with results from every node are not presented for this test as the number of packets expected per node is only one and the table would not present any additional information.

Parameter Name	Value
Power Output	0dBm
Backoff time	30 seconds
Maximum Retransmissions	7
CCR	CCRHZ
Channels	Channels II
No. of Nodes	15
Packets per node	1
Maximum Retrans achieved	7
Network Time	2 minutes

Table 5.8: Parameters and Results for Table Test with fixed number of packets

The final test helped to conclude that the protocol can handle 15 nodes, each trying to send 1 packet if the network runs for 2 minutes. This was an indicator about the robustness of the protocol and the protocols' ability to handle messages from multiple nodes. This delay is a drawback of using asynchronous and sequential collection. There is work that shows that synchronous MAC could be better for dense deployments. However, an asynchronous MAC is better when there are mobile elements in the system that do not necessarily follow a fixed schedule.

Chapter 6

Field Tests

6.1 Introduction

This chapter describes the field tests conducted. The tests were conducted in two different set-ups. The first set of tests were conducted indoors and the UAV-node was moved manually by the author. The second set of tests were conducted outdoors. The tests and their results are described in this chapter. The chapter also presents a conclusion for all the tests performed and the open areas of research that would form an extension to the work described in the thesis.

6.2 Protocols' Evaluation with Moving Node

6.2.1 Tests

The final indoor test conducted was a corridor test. It was a two-fold test with a third test done for comparison with current state-of-the-art solutions in the market. The main aim of the test was to achieve 100% data reliability in all cases. This was the first real-scenario deployment with the UAV-node moving. The UAV-node was held by the author while walking along the length of the corridor. The following parameters were recorded by the UAV-node as part of the tests: RSSI, time, the sequence number of packet, the number of retransmissions, messaging primitive¹ and the transmitter/receiver of the packet.

In the first part of this test four nodes were deployed in the corridor. The test set-up is shown in Figure 6.1. Three out of the four nodes have been circled in the picture. The transmit power of the nodes was such that there was minimal overlapping between two FFDs. In this test, the UAV-node was moved at a constant speed by the author. This emulated the movement of the UAV.

¹Primitives from Section 4.5



Figure 6.1: Corridor Test

In the second part of this test, two parallel rows of 8 nodes and 7 nodes were formed. All of the nodes were at the minimum power level and had to send 1 packet each to the UAV-node. The time required to receive all 15 packets was recorded.

The third part of this test was a test with Collection Tree Protocol (CTP) [33]. This test was used to point to the advantages of using a UAV based collection scheme instead of the traditional routing based data transfer. 11 TelosB nodes were deployed linearly along the length of the corridor for this test. The sink node was placed at either end of this linear topology. All 11 nodes had to send a packet every 30 seconds to the sink. As the nodes were at the lowest power setting of -25dBm, the packets had to be routed to reach the sink.

6.2.2 Results and Analysis

An initial set of tests were performed for the purpose of parameter tuning and its results obtained are presented in Appendix D.1. Once the parameters were fixed, the first test mentioned in Section 6.2.1 was performed and its results are presented in Tables 6.1 and 6.2. In this test, the UAV-node passed by every node 7 times, so

the expected number of transactions per node is 7.

Parameter Name	Value
Power Output	-15dBm
Backoff time	30 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels II
No. of Nodes	4
Packets per node	7
Total number of Transactions	28
Total number of retransmissions	107
Total number of packets	140
Maximum Retrans achieved	15

Table 6.1: Parameters and Results for Corridor test

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	29	7	100%	0.035
Node 2	31	7	100%	0.107
Node 3	32	7	100%	0.142
Node 5	48	7	100%	0.714
Total	140	28		0.25

Table 6.2: Results for Corridor test separated by Node ID

Table 6.2 shows that the data reliability is 100% for all nodes. The protocol works with 4 nodes in the system and churn is in line with the expectation as can be seen in Section 5.5. The next part of the test was conducted to check the robustness of the protocol. In this test, the network time was recorded for 15 nodes each trying to send 1 data packet to the UAV-node. Table 6.3 shows the parameters and results of the test.

Parameter Name	Value
Power Output	-25dBm
Backoff time	30 seconds
Maximum Retransmissions	7
CCR	CCRHZ
Channels	Channels II
No. of Nodes	15
Packets per node	1
Time to Travel Corridor	30 seconds
Maximum Retrans achieved	15
Network Time	2.5 minutes

Table 6.3: Parameters and Results for Corridor test with 15 nodes

Table 6.3 shows that all packets were received in 2.5 minutes. The test helped show protocols' robustness and its ability to handle messages from multiple overlapping nodes.

The motivation for the third test was to do a comparative analysis with a standard multihop scenario. The parameters for CTP are presented in Table 6.4. The packet reception rate for CTP is plotted against the number of hops and is shown in Figure 6.2. It shows that as the number of hops increase, the reliability reduces. Whereas, the data reliability of the protocol in the thesis is 100%.

Parameter Name	Value
Power Output	-25dBm
Report Interval	30 seconds
Maximum Retransmissions	15
Channel	11
No. of Nodes	15
Packets per node	1
Network Time	20 minutes

Table 6.4: Parameters for CTP test

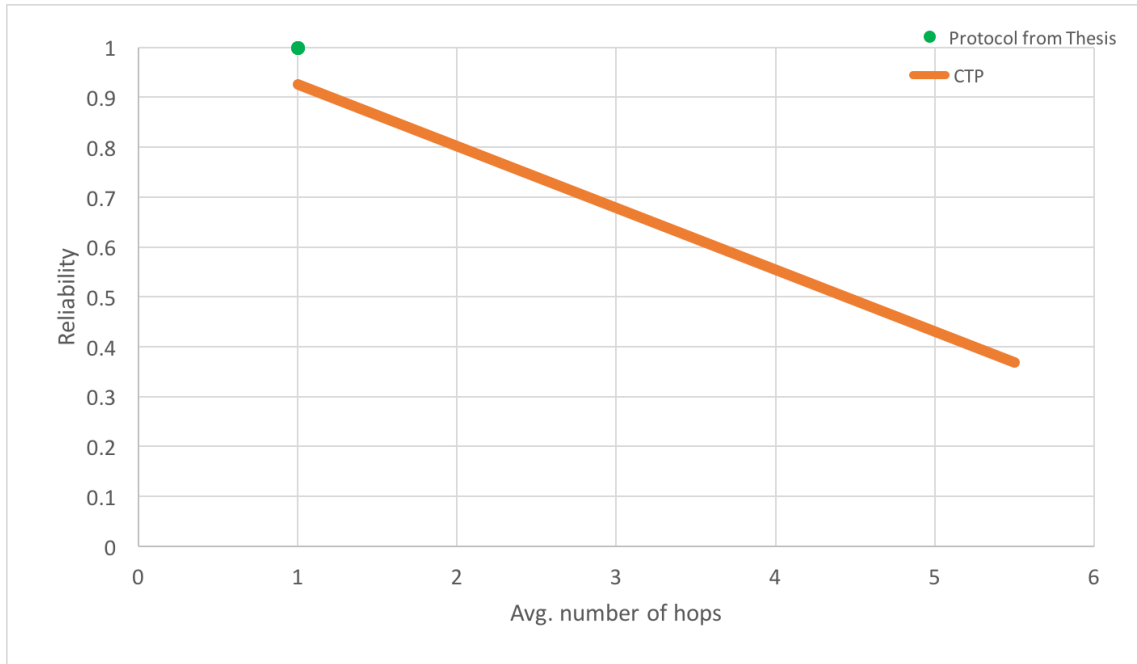


Figure 6.2: Comparison between results from CTP and protocol from thesis

6.3 Outdoor Deployment and Evaluation

6.3.1 Tests

A series of outdoor tests were conducted to validate the protocol. There were a number of requirements for the outdoor location where the tests had to be performed. Firstly, the UAV laws in London are quite stringent and so the location had to meet all of those laws. Secondly, the location had to be an open flat space with a 100m line-of-sight in all four directions from the centre. For this reason, the test location chosen was Hampstead Heath Extension in London. For these tests, three Openmote-CC2538 nodes and DJI Matrice 100 were used. The aim of these tests was to calculate the approximate range that could be covered with the nodes and then verifying the protocol. For all three tests, a sniffer was set-up at point A in Figure 6.3.

In the first part of the test, the UAV-node was moved at a constant speed by the author towards the four points marked in the map starting from point A. The test was conducted to calculate the actual range possible at a height of 1m. The sniffer was logging all broadcast packets sent by the UAV-node and their RSSI values.

In the second part of the test, the UAV-node was attached to the bottom of the UAV. The UAV was then flown from the point A towards the four points marked in Figure 6.3. The broadcast packets were logged by the sniffer along with the RSSI

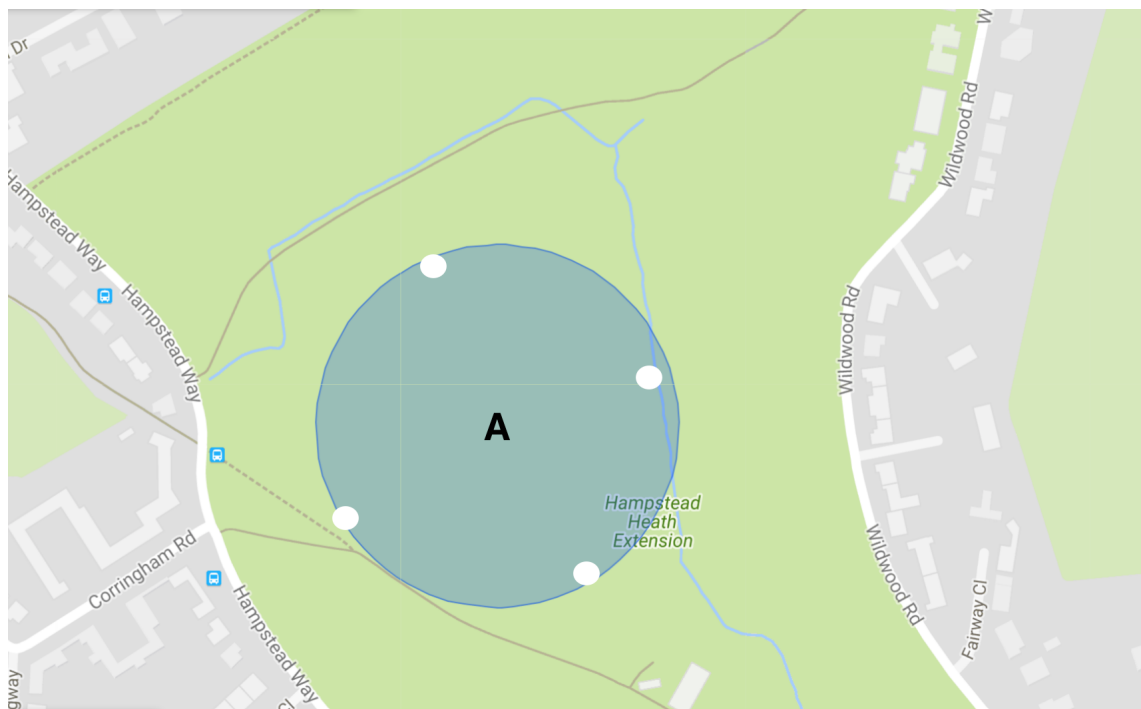


Figure 6.3: Outdoor tests conducted in the marked region with a radius of 100m centred at Lat:51.575036, Long:-0.185947

values. The UAV was flown approximately at a height of 10m.

In the third part of the test, the UAV-node was attached to the bottom of the UAV. The UAV was flown from a stationery state on the ground at a constant speed to a maximum altitude of 10m. The broadcast messages from the UAV-node were recorded by the sniffer on the ground along with the RSSI values.

The final part of the test was conducted to verify the speeds at which the data reliability was a 100% and the protocol functioned as expected. For the test, the FFD was kept next to the sniffer in the same orientation so as to passively listen to the communication between the FFD and UAV-node without interfering with the timing parameters. The UAV was flown at speeds varying from 0-18m/s from one of the four marked points in Figure 6.3 towards point A. As the sniffer could only sense data on Channel 26 i.e. control channel, data was extracted from the flash memory to verify whether the data exchange actually took place.

6.3.2 Results and Analysis

The results from first test are shown in Figure 6.4. The figure shows the change in RSSI values when the UAV-node is moved away from the sniffer and back to it. The node is moved 100m away and back and held at a height of 1m. The plot shows a steady change in the RSSI values with a constant speed of 5m/s. The RSSI value reduces as the node is moved further away and increases as the node is moved closer to the sniffer. The lowest RSSI recorded is -99dBm because of the receiver sensitivity of the CC2538.

Figure 6.5 shows the results of the second test performed when the node was attached to the bottom of the UAV and it was flown at a height of 10m. Figure 6.5 shows the change in RSSI values when the UAV is flown from 150m away towards the sniffer. The plot shows a steep increase in the RSSI values as the UAV-node is flown towards the sniffer. The maximum speed at which the UAV was flown was 18m/s.

Figure 6.6 shows the polar plot for the communication range possible around the centre of Figure 6.3, the outdoor map. The outer plot of Figure 6.6 is the maximum distance achieved due to the lack of space on the field. The plot shows that the communication range possible at an altitude of 1m is lesser than the range possible at an altitude of 10m. It also helps to prove that using a UAV with a wireless communication node would give more time for transactions to complete compared to when the node is closer to the ground (e.g. ground robots).

The results from third part of the test are presented in Figure 6.7. The figure shows the plot when UAV's altitude is varied from 0-10m at a fixed speed. It can be seen that there is no significant change in the RSSI values when the altitude is varied.

The results of the final test are presented in Figure 6.8. The graph shows that

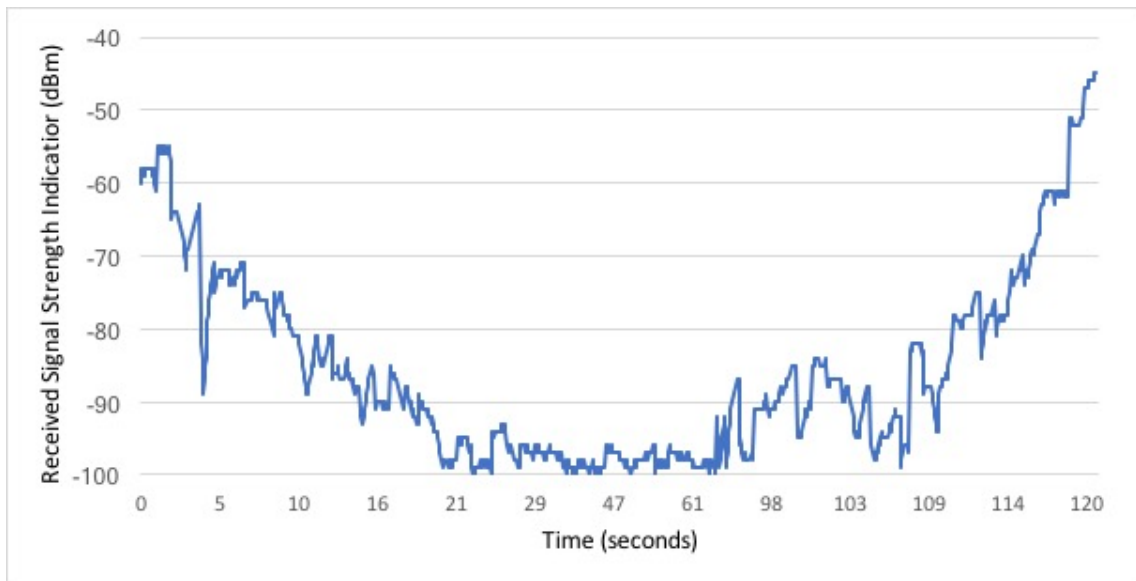


Figure 6.4: RSSI values when the node is held 1m high and is moved away and back from the sniffer

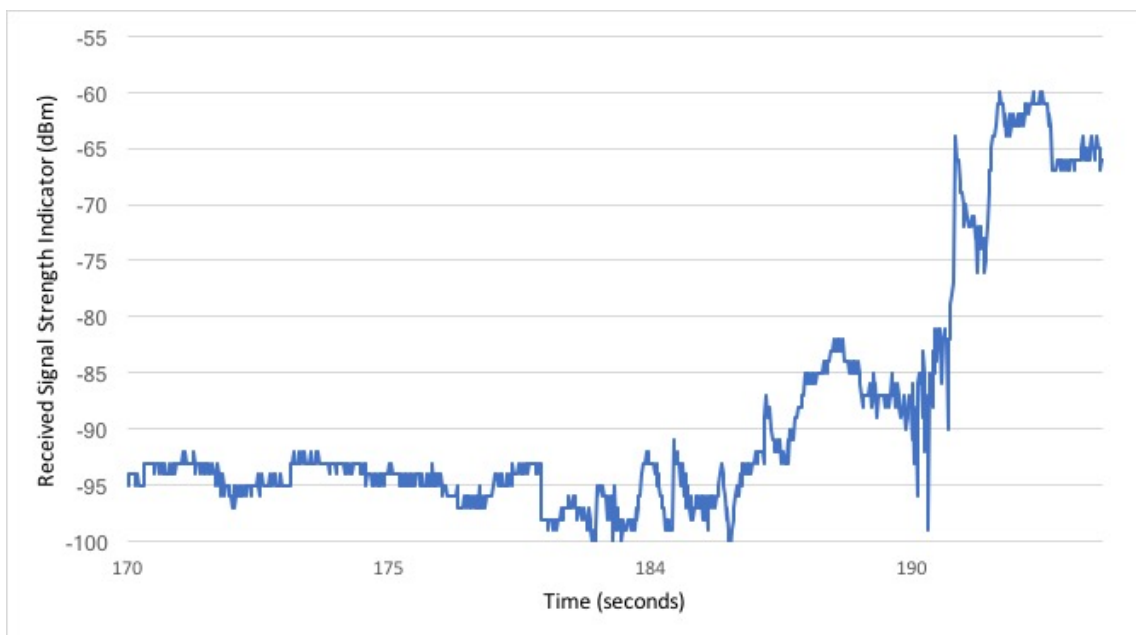


Figure 6.5: RSSI values when the UAV is flown towards the sniffer from 150m away at an altitude of 10m

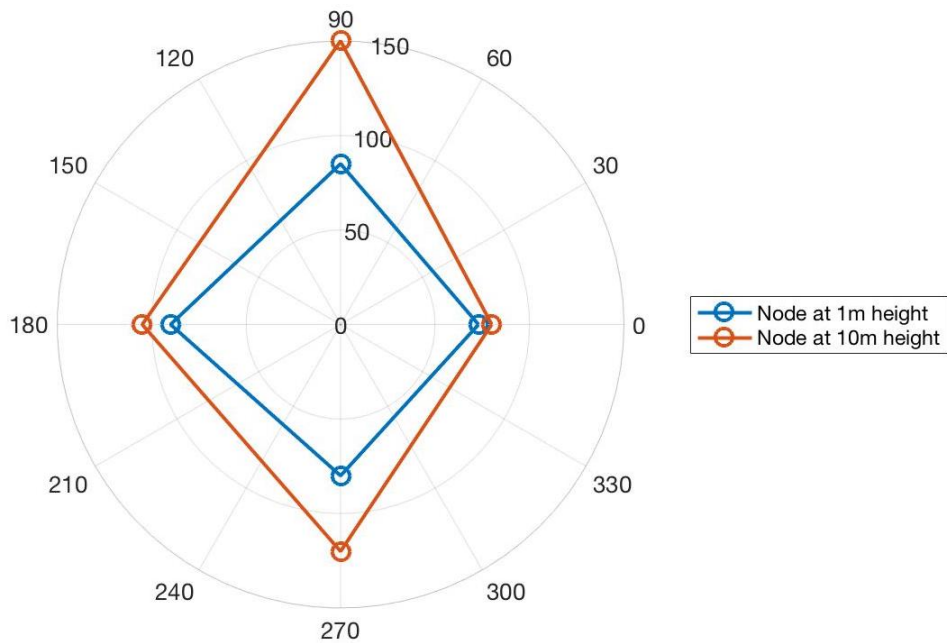


Figure 6.6: Communication range (in m) around the centre of the field at altitudes of 1m and 10m. The results for 1m were the maximum possible and the results for 10m are maximum achievable due to lack of space on the field

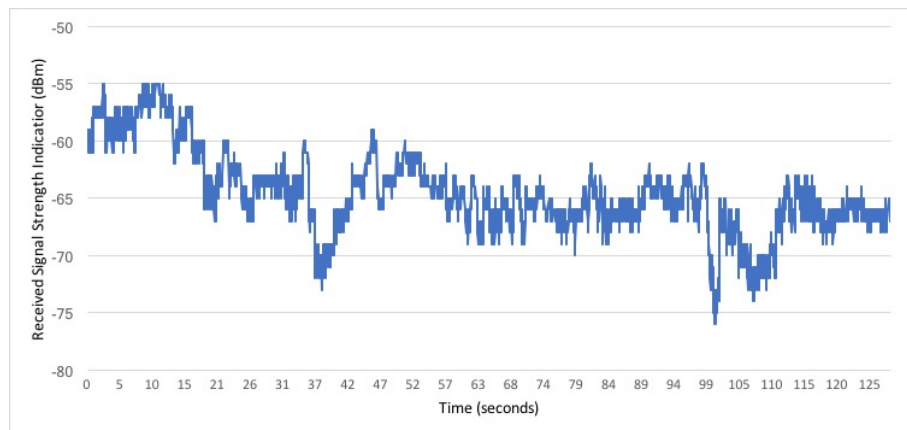


Figure 6.7: RSSI values when the altitude of the UAV is varied from 0 to 10m

the maximum speed at which the Matrice 100 could be flown at was 18m/s. The protocol worked reliably from 0-12m/s. From 12-18m/s the packets were received by the FFD and the sniffer but the REPLY packets were not received or acknowledged by the UAV. One possible reason for this problem could be the shielding effect that is caused by the UAV when it is flying forward. When the UAV flies forward, it tilts in that direction, because of this the broadcast packets could be received by the sniffer but the reply packets were not delivered reliably. As this problem was discovered with field tests, new set of outdoor tests would need to be conducted. The new tests would verify this hypothesis and based on that a solution will be proposed to mitigate this problem.

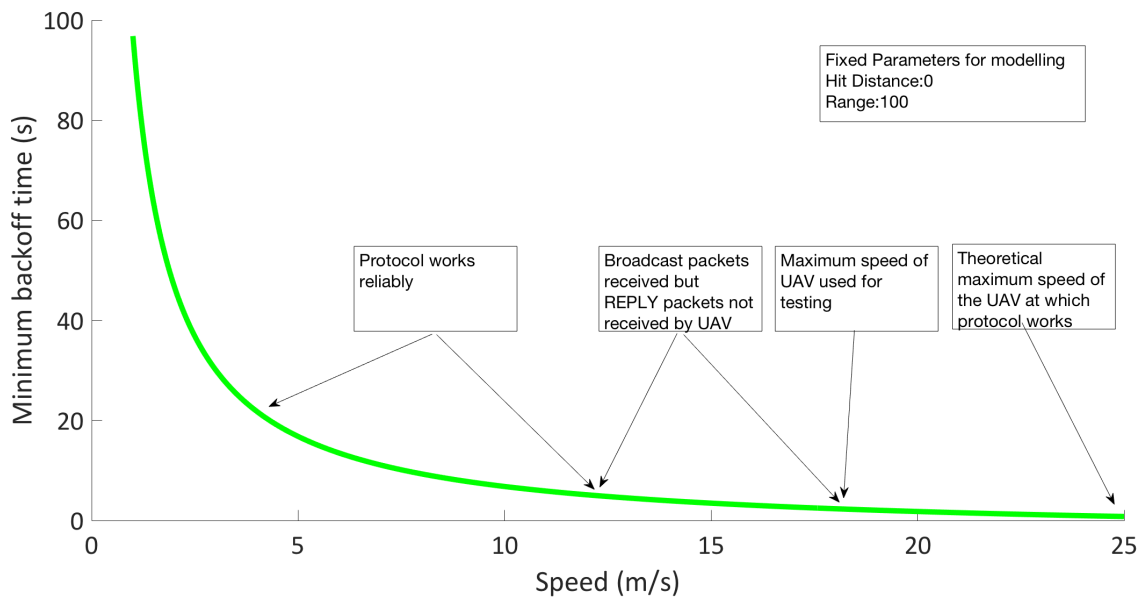


Figure 6.8: Results from field tests overlaid on top of the modelling results

6.4 Requirements Satisfied

The following list highlights the implementation and tests that were performed to show that the requirements mentioned in Section 4.2 are met with the protocol. The list is as follows:

- Work with a mobile aerial sink: Tests performed with UAV prove the protocol works with a mobile aerial sink.
- Have 100% data reliability: Sections 5.5 and 6.2 show a 100% data reliability under reasonable and realistic conditions.
- Data transfer needs to be quick: The entire transaction takes 3 seconds.

- Work reliably in harsh conditions: Tested in humid environments as well as indoors.
- Be robust and scalable: Tests with 15 nodes carried out.
- Be able to handle hidden terminal problems: Channel changing implemented so data transfer takes place on another channel, verified in lab tests.
- Be extensible to enable clustering: Multiple channel check implemented so nodes communicate on a different channel with each other and on another channel with the UAV.
- Be energy efficient: asynchronous MAC reduces the time the nodes are awake to communicate and hence save energy.

6.5 Conclusion

The tests conducted with simulation helped to decide the primitives from Rime stack and calculate the theoretical maximum in terms of speed of the UAV in which there would be a 100% reliable data reception. It was concluded that reliable unicast was better when sending REPLY, ACCEPT, REQSEQ, REPSEQ and BACKOFF whereas unicast was better when sending data packets. This analysis was validated with the tests conducted in the hardware. Simulation in MATLAB helped deduce that with the current timing parameters of the protocol, the UAV could fly at a maximum speed of 25m/s to achieve 100% data reliability with a communication range of 100m between the UAV and FFD.

The hardware tests helped verify the results from simulation. The table tests concluded that the protocol is robust with 15 nodes in communication range of each other and still ensure 100% data reception. The table test was also important in tweaking the parameters of the protocol to ensure that all the data was received successfully during the corridor test. The corridor test was the final test conducted indoors. This test proved that even with a moving node, there was 100% data reception. A comparison with the CTP test was conducted to show that as the distance increases, the number of hops increase and the reliability of CTP reduces.

During the outdoor testing, a number of results were obtained. It was noticed that when there was a change in altitude at a fixed position RSSI values did not deviate a lot. The communication range was more when the altitude of the UAV-node was changed from 1m to 10m. The change in RSSI values was linear when the UAV was flown closer to and away from the sniffer. The protocol developed in the thesis works reliably with a 100% data reliability when the speed of the UAV is between 0 and 12 m/s. Between the speeds of 12-18m/s, shielding effect plays a part and the packets are received reliably by the stationery nodes but are dropped by the moving

UAV-node.

The protocol presented in the thesis is a replacement for scenarios in which multihop routing is required. The packets would not need to be routed as they could be directly sent to the mobile sink. This would also ensure a lower power consumption compared to the multihop case as the nodes would have to be awake for shorter periods of time and would not be responsible for data routing from other nodes.

6.6 Future Areas of Research

As described in Section 3.2, more research needs to be done to make this system field-ready. The next research problem would be to study the shielding effect from the UAVs. This could have a significant impact on the communication between the nodes. Another addition to this system would be to deploy efficient node clustering algorithms. This would help the UAV gather all the data without compromising on its speed. Since the nodes need to be deployed on wind-mills that have an average lifetime of 25 years, wireless power transfer or energy harvesting solutions need to be researched and integrated into the system.

The flight controller of the UAV needs to be integrated with the nodes so that the UAV can make optimum decisions about its path. An autonomous path-planning algorithm needs to be deployed on the UAVs to mitigate the controller problems and reduce human involvement in the system. UAVs could also be designed to meet the specific requirements of the scenario in which they need to be deployed.

Chapter 7

Conclusion

This thesis presents a case for UAVs as mobile sinks in WSN with the possibility to recharge the nodes. The scenarios that this thesis considered were wind-farms and oil/gas pipelines. The protocol presented in the thesis is one possible solution for these scenarios. The state-of-the-art solutions for these scenarios involves connecting a number of gateways on the wind-farm or along the oil/gas pipeline. It also involves time investment by humans for routine monitoring of the systems. Using UAVs is one of the possible optimisation to reduce the human involvement in the system. UAVs along with WSN are a power-efficient replacement for routine monitoring of the wind-farms and oil/gas pipelines.

The specific contribution of this thesis was a communication protocol that ensures a 100% data reliability in scenarios with mobile aerial sinks. The results presented for the protocol show 100% data reliability for a theoretical maximum speed of 25m/s of the UAV.

The outdoor tests showed that a change in altitude had no significant deviation in RSSI but the actual communication range was significantly improved. It helps prove that a UAV with a wireless communication node would give more time for transactions to complete compared to when the node is closer to the ground (e.g. when using ground robots as data sinks). The tests also showed that the protocol works reliably when the speed of UAV is less than 12m/s.

The power analysis done as part of the thesis gives an approximate lifetime of the device. Based on the model developed, the operational lifetime of the nodes can extend to 1.8 years using a typical 2400 mAH battery. It warrants the need to look at energy harvesting solutions or the possibility of inductive wireless power transfer from the UAV to charge the nodes.

This thesis presents a cross layer protocol that has been designed from the bottom-up to ensure a 100% data reliability. It also discusses the power requirements of a typical sensor node and the need for looking into energy harvesting or wireless power transfer solutions to increase the node lifetime. This thesis has opened up a

number of interesting research problems (Described in Section 6.6) and deployment scenarios that were previously unknown. This thesis tries to tackle the problem of communication between a moving sink and static node deployment while trying to calculate the approximate lifetime of the device.

Bibliography

- [1] “WSN White Paper,” Available at <http://www.ni.com/white-paper/7142/en/>.
- [2] “Data centric middleware for the integration of wireless sensor networks and mobile robots,” Available at http://personal.us.es/imaza/papers/conferences/gil_robotica07_web.pdf.
- [3] “Efficient strategies for collecting data from wireless sensor network nodes using mobile robots,” Available at <http://www-users.cs.umn.edu/~isler/pub/isrr09.pdf>.
- [4] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, “Controllably mobile infrastructure for low energy embedded networks,” *IEEE Trans. Mob. Comput.*, vol. 5, no. 8, pp. 958–972, 2006.
- [5] “DJI website,” Available at <http://www.dji.com>.
- [6] “Gartner report on drone market,” Available at <http://www.gartner.com/newsroom/id/3602317>.
- [7] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, “Exploiting sink mobility for maximizing sensor networks lifetime,” in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Jan 2005, pp. 287a–287a.
- [8] S. Sotheara, K. Aso, N. Aomi, and S. Shimamoto, “Effective data gathering and energy efficient communication protocol in wireless sensor networks employing uav,” in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2014, pp. 2342–2347.
- [9] D. T. Ho, E. I. Grøtli, and T. A. Johansen, “Heuristic algorithm and cooperative relay for energy efficient data collection with a uav and wsn,” in *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, Jan 2013, pp. 346–351.
- [10] D.-T. Ho, E. I. Grøtli, P. B. Sujit, T. A. Johansen, and J. B. Sousa, “Optimization of wireless sensor network and uav data acquisition,” *Journal of Intelligent & Robotic Systems*, vol. 78, no. 1, pp. 159–179, 2015.

- [11] M. Dong, K. Ota, M. Lin, Z. Tang, S. Du, and H. Zhu, "Uav-assisted data gathering in wireless sensor networks," *The Journal of Supercomputing*, vol. 70, no. 3, pp. 1142–1155, 2014.
- [12] "Sensors used in wind-mills," Available at <http://www.gemssensors.co.uk/markets/alt-energy>.
- [13] "Wind-farm monitoring zigbee solutions," Available at <http://www.techbase.eu/en/solution/monitoring-and-control/124-wind-farm-monitoring.html>.
- [14] "Rampion offshore wind-farm," Available at <https://www.eonenergy.com/About-eon/our-company/generation/planning-for-the-future/wind/offshore/rampion-offshore-wind-farm>.
- [15] "Rampion offshore wind-farm distance information," Available at <http://www.power-technology.com/projects/rampion-offshore-wind-project-english-channel/>.
- [16] "Sensors used in oil and gas pipelines," Available at <http://www.gemssensors.com/markets/oil-and-gas>.
- [17] "Gas pipelines monitoring with uavs," Available at <http://www.dnaindia.com/money/report-gail-to-start-using-drones-to-secure-gas-pipelines-2202977>.
- [18] "Openmote," More details at <http://openmote.com>.
- [19] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, ser. IPSN '05. Piscataway, NJ, USA: IEEE Press, 2005.
- [20] "DJI Matrice 100," Description at <https://www.dji.com/matrice100>.
- [21] "DJI Lightbridge Controller," Available at <http://www.dji.com/lightbridge-2>.
- [22] "Riot OS," Description at <http://riot-os.org>.
- [23] "TinyOS," Description at <https://github.com/tinyos>.
- [24] "ConikiOS," Description at <http://www.contiki-os.org>.
- [25] "Free RTOS," Description at <http://www.freertos.org>.
- [26] "OSI Model's seven layers," Available at <https://support.microsoft.com/en-us/help/103884/the-osi-model-s-seven-layers-defined-and-functions-explained>.
- [27] "Mobility plugin used in cooja," Available at <https://sourceforge.net/p/contiki/projects/code/HEAD/tree/sics.se/mobility/>.
- [28] "ContikiMAC," Available at <http://soda.swedish-ict.se/5128/1/contikimac-report.pdf>.

- [29] “Time Slotted Channel Hopping RFC,” Available at <https://tools.ietf.org/html/rfc7554>.
- [30] M. Barcelo, A. Correa, X. Vilajosana, J. L. Vicario, and A. Morell, “Novel routing approach for the tsch mode of ieee 802.15.14e in wireless sensor networks with mobile nodes,” in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, Sept 2014, pp. 1–5.
- [31] “Coffee File System in ConikiOS,” Description at <https://github.com/contiki-os/contiki/wiki/Coffee-file-system-example>.
- [32] “CC2538 Datasheet,” Available at <http://www.ti.com/lit/ds/symlink/cc2538.pdf>.
- [33] “CTP: Collection Tree Protocol,” Available at <https://sing.stanford.edu/gnawali/ctp/>.

Appendix A

UAV modifications

DJI Matrice 100 uses a custom DJI Lightbridge protocol to communicate. This protocol operates in the 2.4GHz spectrum. However, the bandwidth and power levels for the channels are different from traditional IEEE 802.15.4 or 802.11 channels. Table A.1 describes the channels along with their lower, centre and upper frequencies. It also correlates the channels to the ones describes in IEEE 802.11 standard.

Lightbridge Channel	Lower Frequency	Centre Frequency	Upper Frequency	WiFi Channels
13	2401	2406	2411	1
14	2411	2416	2421	1,2,3
15	2421	2426	2431	2,3,4,5
16	2431	2436	2441	4,5,6,7
17	2441	2446	2451	6,7,8,9
18	2451	2456	2461	8,9,10,11
19	2461	2466	2471	10,11,12,13
20	2471	2476	2481	12,13,14

Table A.1: DJI Lightbridge Frequency Division

The power output of the controller is 20dBm for the channel in which it communicating with the UAV. The controller also tries to check if there is any other better channel available by sending a check packet on other channels at maximum power. This leads to excessive amounts of energy in the spectrum at regular intervals and disrupts the entire 2.4GHz ISM band. In Figure A.3, region A shows the 2.4GHz spectrum when the controller keeps checking for free channels. This plot is a max-hold plot of the spectrum. To overcome the problem of periodic bursts of energy in the spectrum, a reliable unicast was used instead of a normal unicast.

Even after forcing the DJI controller on a channel, if the channel was too close to the channels on which the protocol was communicating there was still packet loss. This packet loss was due to the fact that the controller was broadcasting data at 20dBm whereas the nodes broadcast data at maximum 7dBm. In Figure A.1, region A shows the max-hold plot for the 2.4GHz spectrum when the nodes broadcast data

at Channel 26. It was concluded that it would be better to move just outside the ISM band such that there was no noise in the channel. Doing this workaround is justified as the final system that is envisioned could be completely autonomous and would not need to communicate with a controller and so the channels would be free by default. Another potential solution could be that the sensors communicated on a different part of the spectrum (e.g. $<1\text{GHz}$).

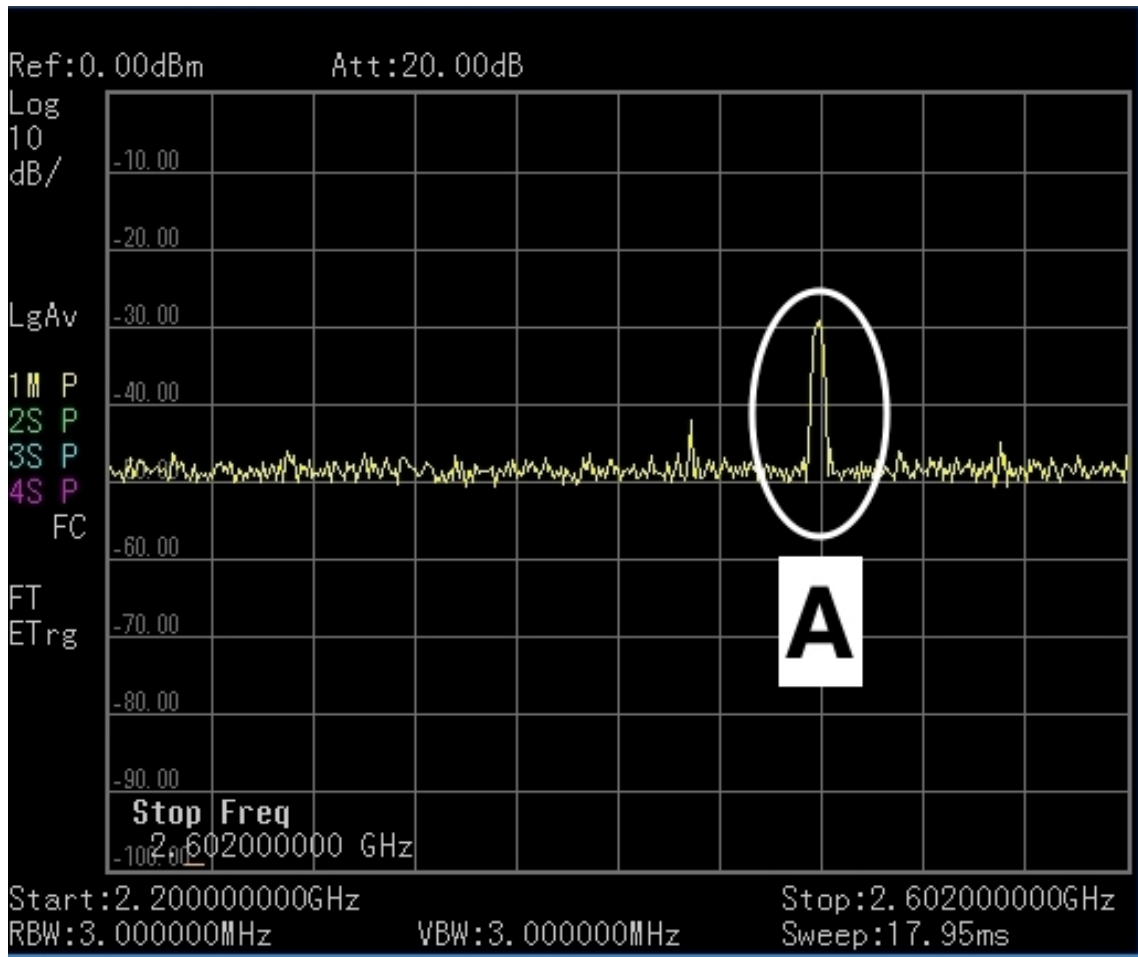


Figure A.1: IEEE 802.15.4 Channel 26 in 2.4GHz spectrum at 7dBm output power

To force the drone to use a channel outside the ISM band, the following data was added to the data folder of the application on the phone. The files' data is shown in Figure A.2. The file shows that the code to choose channels out of the ISM Band already exists in the application files and just needed to be enabled for the purpose of this thesis. In Figure A.3, region B shows the max-hold plot of the DJI controller communicating outside the 2.4GHz ISM band while still polling the channels in the ISM band. The controller and the UAV are communicating at the centre frequency of 2.596GHz with a 10MHz bandwidth.


```

00000000: |444a 4920 534f 4654 5741 5245 2043 4f4e 4649 4720 3a20 444f :DJI SOFTWARE CONFIG : DO
00000018: 204e 4f54 2044 454c 4554 4521 0000 0001 1111 1111 1111 1111 : NOT DELETE!.....

```

Figure A.2: DJI Config File

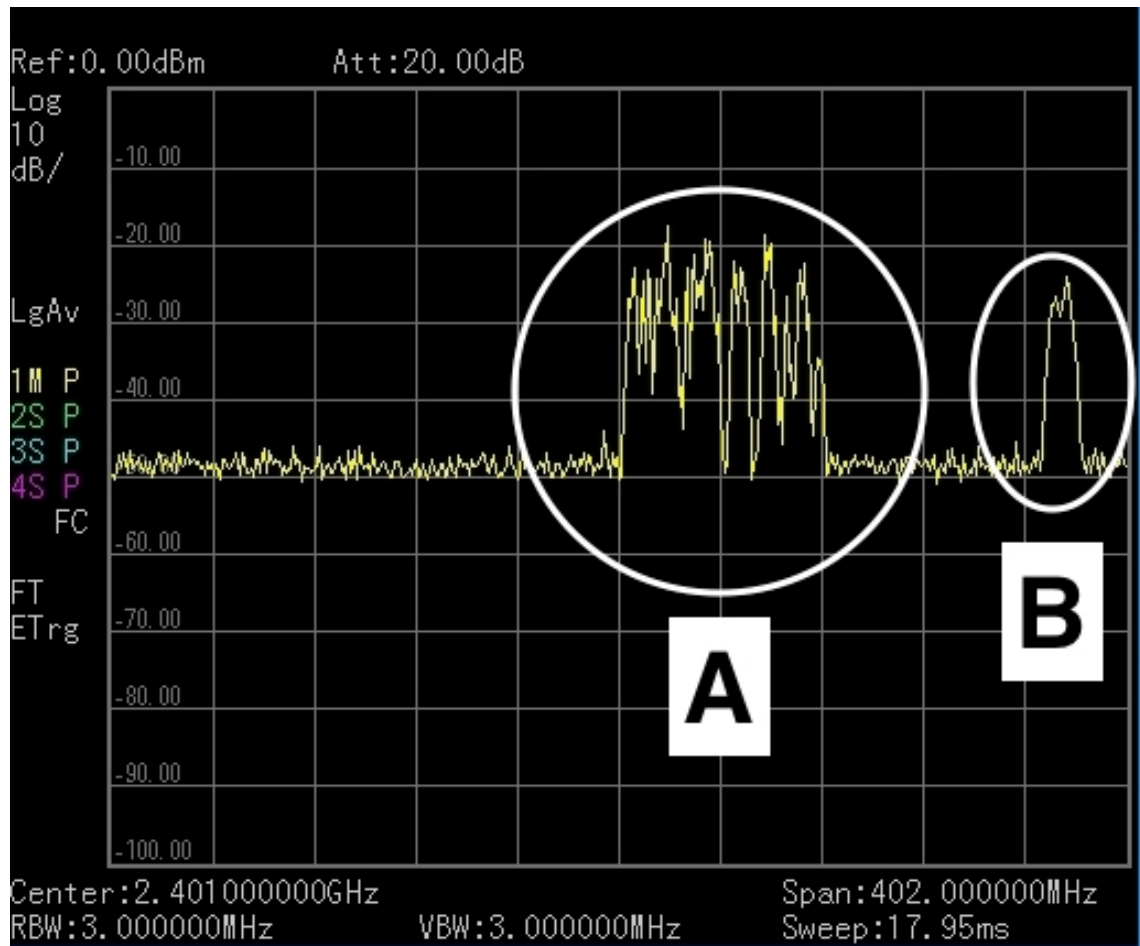


Figure A.3: Region A - DJI Controller communicating in ISM band, Region B - DJI Controller communicating outside ISM Band i.e. modified channel

Appendix B

Protocol Diagram with Hidden UAV node

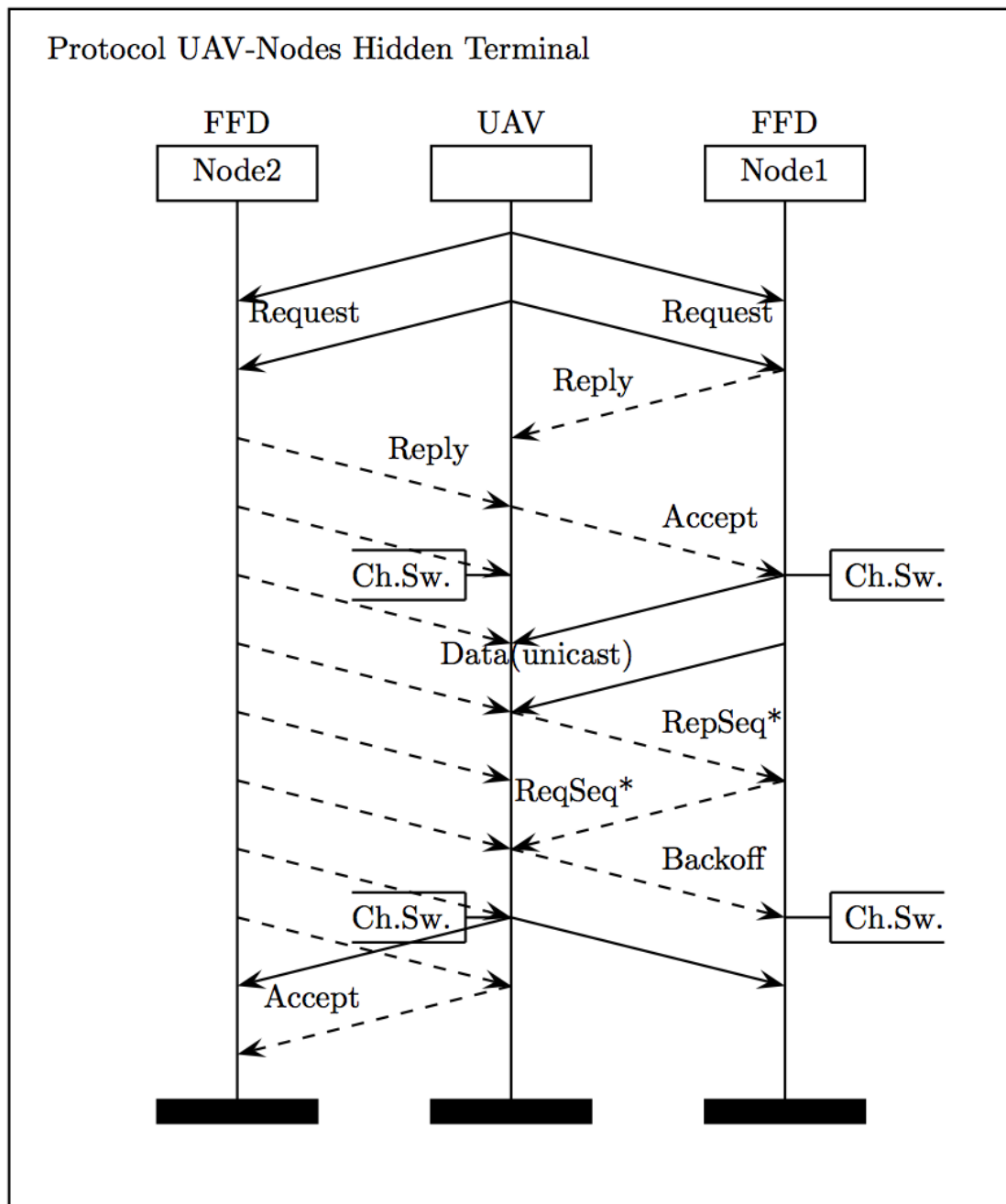


Figure B.1: Protocol Diagram with two nodes

Appendix C

Mathematical Models

This appendix describes the Mathematical model developed to calculate the current consumption by the protocol and the relation between speed and reliability.

C.1 Current Consumption Model

The mathematical model created for the current consumption was programmed in MATLAB and then compared with the results obtained from hardware. The model was tested in simulation and hardware to ensure its correctness. The most appropriate way to calculate the lifetime of the node is to measure the current it consumes for every operation and compare this value with the mAH rating of the battery. For this reason, the model calculated the approximate current consumption by different operations of the node. The average current values for the model were from the CC2538 datasheet [32]. The datasheet also had the time taken for different operations, e.g. switching between RX/TX, and startup time. Other timing parameters like the sleep time, time to send a packet and channel check rates were determined by the protocol. The current consumptions and their timings from datasheet are presented in Table C.1. The parameters are used to obtain results shown in Figure 5.8 and Figure 5.12.

Parameter Name	Current Needed (mA)	Average time (s)
Start MCU	10	0.0340
Start Radio	15	0.05
Switch RX/TX	1	0.03
Power mode 2 (IDLE)	0.6	Protocol Specific
Send 1 packet	24	Protocol Specific
Receive packet	20	Protocol Specific

Table C.1: Current and timings used for modelling of Current consumption

The average current required to run a protocol once was calculated with the following equation:

$$I_{avg} = (\sum_{t=0}^{t=T} I_t * t) / (T)$$

where T is the total time in seconds for one transaction to complete and I_t is the instantaneous current for the current operation being performed.

The average current needed for 1 hour is calculated as follows:

$$I_h = I_{avg} * (T/3600) + I_{sleep} * (1 - T)$$

where I_{sleep} is the current when the node is sleeping.

C.2 Speed vs Reliability Model

A mathematical model to calculate the theoretical maximum while still achieving a 100% reliability is formulated in the following section. One of the important requirement of the protocol is for the drone to travel at the highest speed possible to cover as much distance as possible. To calculate the probability of success, free time or backoff time is used as the parameter. Free/Backoff time is calculated with the following equation:

$$t = ((r - d)/v) - T$$

where t is the time remaining in seconds, r is the range of communication between FFD and UAV in meters, d is the distance in meters at which FFD receives REQUEST packet, v is the speed in meters/second of the UAV and T is the time in seconds required to complete one transaction.

Figure 5.14 shows the output of varying v keeping the other parameters fixed. Test varying d was also performed, however only result for varying speed have been shown as it is a deterministic parameter compared to d .

Appendix D

Methods to approach 100% reliable, robust and handling hidden terminal issues

This appendix describes the changes that were made to the protocol to approach a 100% data reliability.

The first protocol that was written used only broadcast and unicast from the Rime communication stack. However, after the first time the UAV was introduced in the system, the problem described in Appendix A was discovered. The first fix to that problem was to force the UAV to communicate out of the ISM band. Besides that, another modification made to the system was introduce reliable unicast instead of unicast. The final protocol that was tested is described in Section 4.5.

The next set of tests conducted were a mixture of corridor and table tests. A number of tests were conducted in a logical sequence. The text below is in the same flow instead of two separate sections as described in Section 5.5 and 6.2. The tests are described sequentially to highlight the changes that were made to achieve the end result of a 100% reliable data reception.

D.1 Reliability Tests

The first run of tests that were conducted after modifying the protocol were the corridor tests. This was to try and simulate the moving UAV-node scenario from the tests that were conducted outdoors. Tables D.1 and D.2 are the results from the first test.

Parameter Name	Value
Network Runtime	3.5 minutes
Power Output	-7dBm
Backoff time	10 seconds
Maximum Retransmissions	31
CCR	CCRHZ
Channels	Channels I
No. of Nodes	5
Packets per node	20
Total number of Transactions	51
Total number of retransmissions	123
Total number of packets	243
Maximum Retrans achieved	4

Table D.1: Parameters and Results for Corridor Test 1

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	76	18	90%	0.0556
Node 2	71	16	80%	0.109
Node 3	36	6	30%	0.5
Node 4	22	3	15 %	0.833
Node 5	38	8	40 %	0.187
Total	243	51		0.191

Table D.2: Results for Corridor Test 1 by node ID

The results show that the reliability was never 100%. The reliability was particularly less with the TelsoB nodes. The analysis concluded that when TelosB was channel hopping between channels 11 and 26, it was dropping a lot of packets. So in the next test, none of the TelosB nodes were channel hopping and they were checking only channel 26. The results of this test are shown in Tables D.3 and D.4.

Parameter Name	Value
Network Runtime	3.5 minutes
Power Output	0dBm
Backoff time	10 seconds
Maximum Retransmissions	31
CCR	CCRHZ
Channels	Channels II
No. of Nodes	5
Packets per node	20
Total number of Transactions	41
Total number of retransmissions	113
Total number of packets	188
Maximum Retrans achieved	4

Table D.3: Parameters and Results for Corridor Test 2

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	52	12	60 %	0.0833
Node 2	49	12	60 %	0.0208
Node 3	48	12	60 %	0
Node 4	19	4	20 %	0.187
Node 5	20	1	5 %	4
Total	188	41		0.146

Table D.4: Results for Corridor Test 2 by node ID

The results from above tables show that although the reliability for two of the TelosB nodes increased, it fell considerably for the third one. The reliability of the Openmote nodes also reduced. The first intuition was that there was radio interference. The test was conducted again by changing the node positions to investigate the problem. The results were similar to the previous test, and it was concluded that there was no external radio interference interfering with the system. However, after analysing the logs, a faulty node in the system was identified. It was a "babbling-idiot" problem, and it was mitigated by eliminating the node from the system. Another problem noticed when running this test was that the UAV-node restarted twice. This problem was linked with the way the flash memory was handled by the protocol. This was fixed and the problem did not appear again.

The next test conducted was with 4 nodes and removing the node with the "babbling-idiot" problem from the network. When the logs were analysed, it was noticed none of the nodes timed out and a lot of time was wasted in retransmitting the same packet by the UAV even when the node was out of communication range. So the maximum number of retransmissions was changed. In earlier tests, the max-

imum number of retransmissions was 31 and it was changed to 15 for the next set of tests. With the reliable unicast sending a retransmit every second, the packet had to be delivered in 16 seconds instead of the the previous tests in which it was retransmitting for 32 seconds. Tables D.5 and D.6 show the results of this test.

Parameter Name	Value
Network Runtime	3.5 minutes
Power Output	-15dBm
Backoff time	10 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels II
No. of Nodes	4
Packets per node	12
Total number of Transactions	16
Total number of retransmissions	53
Total number of packets	72
Maximum Retrans achieved	15

Table D.5: Parameters and Results for Corridor Test 3

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	19	4	33.33 %	0.187
Node 2	30	7	58.33 %	0.0714
Node 3	15	3	25 %	0.25
Node 5	8	2	16.67 %	0
Total	72	16		0.125

Table D.6: Results for Corridor Test 3 by node ID

The results from above table show better results. However, a 100% reliability was still not achieved. Even though the power level was low; the backoff time was too low. Because of that, the Openmote nodes would get up after the backoff time, and the UAV-node would communicate with them even if it did not complete the communication with other nodes. So, the next parameter that was tuned to achieve a 100% reliability was the backoff time. The time to walk the length of the corridor once was 30 seconds. So, the backoff time was changed from 10 seconds to 30 seconds. This ensured that the number of packets expected would coincide with the number of times the UAV-node passed-by the FFDs. The results of this test are shown in Tables D.7 and D.8.

Parameter Name	Value
Power Output	-15dBm
Backoff time	30 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels II
No. of Nodes	4
Packets per node	7
Total number of Transactions	17
Total number of retransmissions	54
Total number of packets	74
Maximum Retrans achieved	5

Table D.7: Parameters and Results for Corridor Test 4

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	20	5	71.43 %	0
Node 2	21	5	71.43 %	0.05
Node 3	16	4	57.14 %	0
Node 5	17	3	42.85%	0.417
Total	74	17		0.088

Table D.8: Results for Corridor Test 4 by node ID

The above test result showed promising results and the test was conducted again without changing the parameters. The results of that test were the best results that were obtained and are presented in Tables 6.1 and 6.2.

D.2 Multiple Nodes/Hidden Terminal Tests

Once the protocol showed 100% data reliability for four nodes, the problem of the hidden node and multiple nodes in the same communication area had to be tested. These tests were also conducted to see maximum packets that could be received with a fixed network time.

The first set of tests conducted had the same parameters as the ones used in the last corridor test. The parameters and results of this test are shown in Tables D.9 and D.10.

Parameter Name	Value
Network Runtime	29 minutes
Power Output	-15dBm
Backoff time	30 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels II
No. of Nodes	4
Packets per node	60/59
Total number of Transactions	238
Total number of retransmissions	523
Total number of packets	984
Maximum Retrans achieved	4

Table D.9: Parameters and Results for Table Test 1

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	248	60	100 %	0.033
Node 2	246	59	100 %	0.042
Node 3	243	60	100 %	0.013
Node 5	247	59	100 %	0.047
Total	984	238		0.034

Table D.10: Results for Table Test 1 by node ID

The results from above table were obtained in the first run of the test without the interference from the controller. The latter two runs had <100% data reliability with the controller trying to communicate with the UAV. When the UAV and controller were trying to communicate, there was high energy on the network as described in Appendix A. The results from one of the runs are shown in Tables D.11 and D.12.

Parameter Name	Value
Network Time	30 minutes
Power Output	-15dBm
Backoff time	30 seconds
Maximum Retransmissions	15
CCR	CCRHZ
Channels	Channels II
No. of Nodes	5
Packets per node	60
Total number of Transactions	259
Total number of retransmissions	603
Total number of packets	1077
Maximum Retrans achieved	4

Table D.11: Parameters and Results for Table Test 2

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
Node 1	211	52	86.67 %	0.0144
Node 2	209	51	85 %	0.0245
Node 3	225	54	90 %	0.0417
Node 5	241	56	93.33 %	0.0759
Node 6	191	46	76.67 %	0.038
Total	1077	259		0.0396

Table D.12: Results for Table Test 2 by node ID

The results from the above test showed that there was a need to run the network for a longer time. Before that could be done, another test with a 15 network time was conducted. The results of the same are presented in Tables 5.4 and 5.5. The results of that test were used as a benchmark, and the test was conducted again to try and achieve a 100% data reliability by extending the network time. The results from ensuring fixed number of packets were delivered and recording network time are presented in Tables 5.6 and 5.7.

D.3 Robustness Test

The above tests proved that the protocol can handle multiple nodes and given enough time achieve a 100% reliability. However, as the environment can be harsh in the cases where this deployment is done, the protocol needs to be robust. One of the ways to check the robustness of the protocol was to force the controller to communicate within the 2.4GHz ISM spectrum and in a channel close to the ones in which the nodes communicate. This would lead to a significantly higher number of

retransmissions and a larger time to achieve the 100% data reliability.

Tables D.13 and D.14 show the results from one of the tests in which the controller was communicating in a channel close to the nodes. The number of retransmissions timed out 3 times. After running the network for 15 minutes, the average number of data packets was 77%. However, running the network for an additional 4.35 minutes ensured that all packets were received successfully.

Parameter Name	Value
Network Runtime	19.35 minutes
Power Output	7dBm(Openmote), 0dBm(TelosB)
Backoff time	30 seconds
Maximum Retransmissions	7
CCR	CCRHZ
Channels	Channels II
No. of Nodes	4
Packets per node	30
Total number of Transactions	120
Total number of retransmissions	383
Total number of packets	591
Maximum Retrans achieved	7
Number of Timeouts	3

Table D.13: Parameters and Results for Table Test 3

Node number	Packets Received	Successful Transactions	Success Ratio	Churn
15 minutes elapsed				
Node 1	116	23	76.67 %	0.261
Node 2	117	24	80 %	0.219
Node 6	119	22	73.33 %	0.432
Node 9	126	24	80 %	0.240
19.35 minutes elapsed				
Node 1	145	30	100 %	0.208
Node 2	141	30	100 %	0.175
Node 6	160	30	100 %	0.333
Node 9	145	30	100 %	0.208
Total	591	120		0.231

Table D.14: Results for Table Test 3 by node ID

Appendix E

Resources

E.1 Code

The code for the thesis can be downloaded from: Link[<https://goo.gl/kYucnC>]

Code is structured as follows:

- `contiki_csma_9` : This folder contains the code for the protocol before the first outdoor tests were performed
- `final_protocol` : This folder contains the code for the final protocol implementation (Refer to Section 4.5 for entire protocol description)
- `power-profile` : This folder contains the custom MAC model developed to verify the Mathematical model for the current consumption (Refer to Section 5.4.1.1 for the MAC description)
- `contiki.patch` : This is a patch that needs to be applied to contiki to run the final working protocol (Should be applied with the following command: `'git apply contiki.path'`)

E.2 Data

The raw data collected from all the tests are available at: Link [<https://goo.gl/kwenXv>]

The folder is structured as follows:

- `collect`: Results of the CTP protocol (Tests and results in Section 6.2)
- `fix_number_of_packets`: Results from running the final protocol for a fixed number of packets in the corridor test (Tests and results described in Section 6.2)

-
- `fix_number_packets_time_to_completion`: Results from running the final protocol until completion for a fixed number of packets (Tests and results described in Section 5.5)
 - `fix_time_tests`: Results from running the protocol for a fixed period of time (Tests and results described in Sections 5.5,6.2)
 - `outdoortest`: Results from the final outdoor tests performed (Tests and results described in Section 6.3)
 - `parameter_tuning_tests`: Results used to tweak the parameters of the protocol (Tests and results described in Section 5.3)
 - `spectrumanalyser_tabletest`: Results from the spectrum analyser